WARUM DAS HOBBY EINEN COMMODORE COMPUTER BRAUCHT.

Weil er dem Hobby-Sammler z. B. die ganze Kollektion katalogisiert.

Weil für den Musik-Fan ein leistungsstarker Synthesizer drin ist: 3 Stimmen mit je 8 Oktaven für heiße oder klassische Concertos von Commodore.

Weil er dem Hobby-Programmierer volle 38 KB RAM für BASIC-Programme und Daten bereitstellt. Oder satte 52 KB für Maschinensprache-Profis.

Weil der Commodore Heimcomputer natürlich auch die hochauflösende Farb-Grafik beherrscht. Samt Sprites für die Spielemacher.

Und weil der Spitzenreiter unter den Heimcomputern ein tüchtiger Mitstreiter bei jeder Art von Papierkrieg ist.

Darum braucht vielleicht nicht nur das Hobby einen Commodore Computer.

Beim Commodore-Vertragshandel, in führenden Warenhäusern, guten Rundfunk-, Fernseh- und Fotofachgeschäften und großen Versandhäusern.

Mehr Information und die Anschrift Ihres nächstgelegenen Commodore-Fachhändlers von: Commodore Büromaschinen GmbH, Abt. MK, Lyoner Str. 38, 6000 Frankfurt/M. 71. Oder per Telefon: Düsseldorf (0211) 3120 47/48 Frankfurt (069) 663 8199 · Hamburg (040) 211386 · München (089) 463009 · Stuttgart (0711) 247329 Basel (061) 237800 · Wien (0222) 675600.

Unsere BTX-Leitseite * 18919 #.

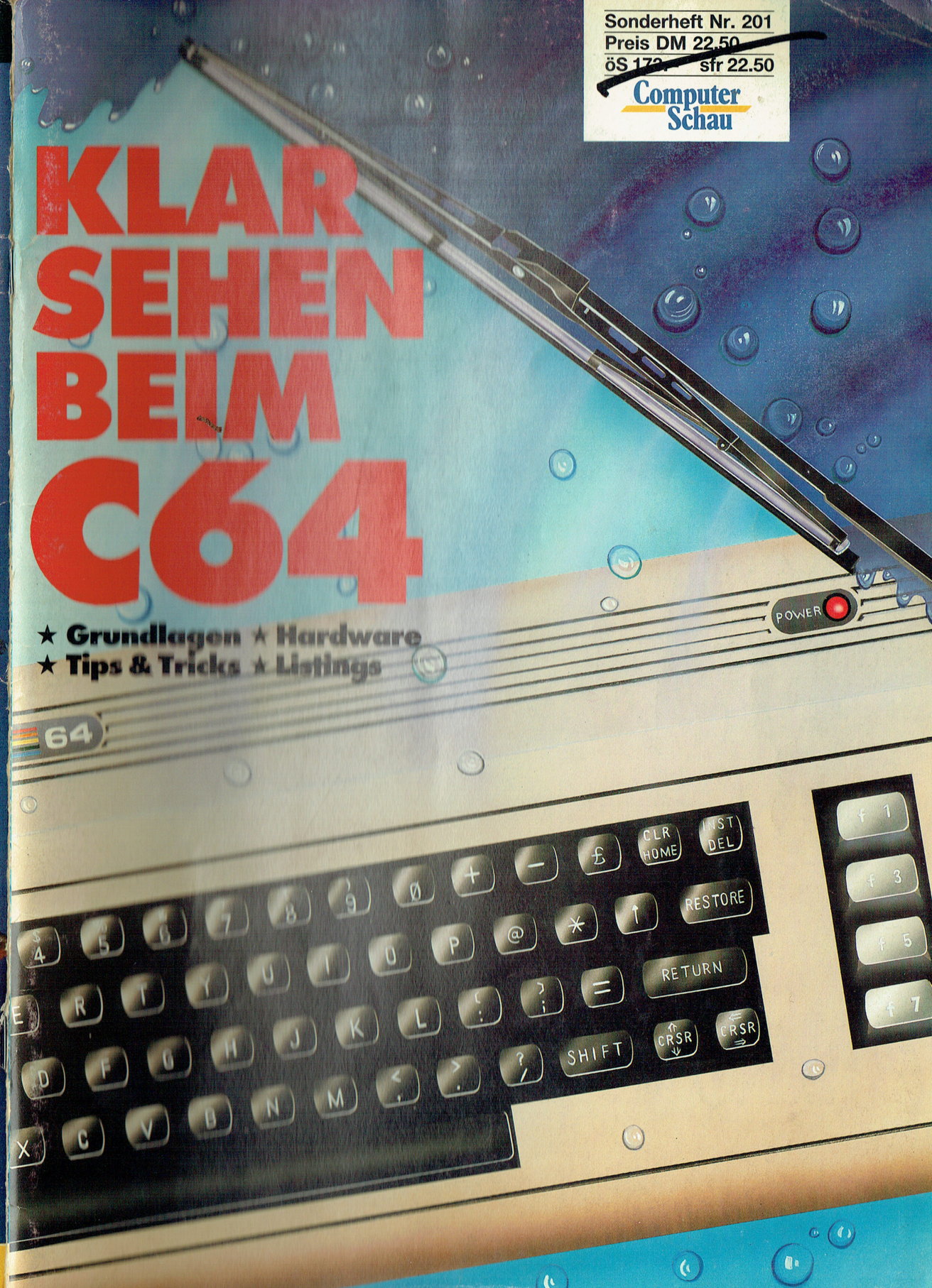
Commodore

Eine gute Idee nach der anderen.

Sonderheft Nr. 201
Preis DM 22.50
öS 17[illegible] sfr 22.50
Computer Schau

KLAR SEHEN BEIM C64

★ Grundlagen ★ Hardware
★ Tips & Tricks ★ Listings



GÖRLITZ COMPUTERBAU

So bestellen Sie:

für Commodore C-64

mit Kabel direkt zum User-Port, steckfertig, Ansprache als Gerät 2 über die RS-232-Schnittstelle. Im Drucker ist die Groß- und Kleinschreibung passend zur CBM-Codierung einstellbar.
Best.-Nr.: 8570

für EPSON HX-20

mit Kabel zur RS-232-Schnittstelle, steckfertig. Besonders interessant für die mobile Korrespondenz: Durch Batterien alles tragbar! Der ASCII-Satz des HX-20 mit deutschen Umlauten ist im Drucker einstellbar, damit auch passend für andere Rechner mit deutschen Umlauten.
Best.-Nr. 8571

für SINCLAIR ZX-Spectrum

(ausgerüstet mit ZX-Interface-1) direkt steckfertig mit Anschlußkabel und 9poligem Stecker an die RS-232-Schnittstelle.
Best.-Nr.: 8572

Für andere Rechner mit einseitig offenem Spezialkabel und Anleitung zum Anschluß.
Best.-Nr.: 8573

Der Einsteiger-Drucker

GÖRLITZ COMPUTERBAU, Hersteller der bekannten Interfaces, bringt einen Drucker, den sich jeder leisten kann.

Schreibmaschine und Drucker in einem, tragbar, gespeist von Batterie, Akku oder Netzteil. Und weil GÖRLITZ für Interfaces bekannt ist, mit einer Rechnerschnittstelle, die Ihre Wünsche erfüllt: Der Zeichenvorrat des Druckers (ASCII mit allen internationalen Umlauten) ist in acht verschiedenen Zeichentabellen vorhanden. Damit drucken Sie sowohl ASCII als auch deutsche Umlaute, passend an C-64, EPSON HX-20, SINCLAIR ZX-SPECTRUM und alle anderen Rechner mit serieller Schnittstelle. Die Druckgeschwindigkeit beträgt 600 Baud, der Druck geschieht über Farbband auf Normalpapier oder ohne Farbband auf Thermopapier, beides fast geräuschlos. Lieferung mit Farbbändern und Anschlußkabel, ohne Batterien.

Zum Einsteiger-Preis von 398,– DM

Hersteller weltweit bekannter Baugruppen, Ingenieurbüro für technisch-wissenschaftliche Sonderanfertigungen, autorisierter EPSON-Händler: Görlitz Computerbau – ... denn andere verkaufen nur!

Ihr Commodore 64 kann viel –
mit unseren Görlitz-Interfaces noch viel mehr!

das bewährte VC-EPSON-INTERFACE:
tausendfach verkauft, europaweit erprobt, bringt alle CBM-Grafikzeichen in vierzig verschiedenen Schriftarten und -breiten, HARDCOPY mit SIMON'S BASIC, sauber und hochauflösend, hat einen eingebauten Selbsttest, einen eigenen Z80-Mikroprozessor, einen 2K-Pufferspeicher. Alle ursprünglichen Funktionen des Druckers bleiben durch das Interface unverändert.
Best.-Nr.: 8422

das neue VC-EPSON-INTERFACE:
bekannte Funktionsvielfalt: neben der EPSON-Schrift sämtliche Originalzeichen der CBM-Rechner in Groß-, Kleinschrift und Grafikdarstellung in 8x8-Matrix, selbsttätige Aufbereitung durch das Interface und Übertragung an den Drucker im Grafikmodus. Neu an diesem VCEI: der 8K-Byte-Pufferspeicher. Damit Sie nie mehr warten müssen. Best.-Nr.: 8424

das vielseitige VC-EPSON-INTERFACE:
wie immer Ihr Drucker heißt, unser VC-Centronics-Interface verbindet ihn komfortabel mit Commodore 64, einzige Voraussetzung: Centronics-Schnittstelle und EPSON-kompatiblen Steuerzeichensatz. Dieses Interface bieten Ihnen optimalen Anschluß Ihres Commodore 64 an preiswerte Drucker. Lieferung der externen Version im Gehäuse. Best.-Nr.: 8423

Die besten Commodore-Interfaces für Drucker

8415	SUPER EPSON INTERFACE (8032/96 usw. an EPSON-Drucker)	450,30
8422	VC-EPSON-Interface (VCEI) zwischen C-64 und EPSON	336,30
8423	VC-Centronics-Interface (VCCI) extern, für STAR u. a.	336,30
8424	VC-EPSON-Interface (VCEI) mit 8K-Puffer	450,30
8490	Drucker RX-80 mit VCEI 8422	1330,00
8491	Drucker RX-80 F/T mit VCEI 8422	1502,00
8493	Drucker FX-80 mit VCEI 8422	1889,00
8494	Drucker FX-100 mit VCEI 8422	2362,00
	I/O-TOOL: zwei RS-232-Schnittstellen zum Einbau in CBM 8032 usw.	
8505	I/O-TOOL RS-232/2 CBM 3000	655,50
8506	I/O-TOOL RS-232/2 CBM 4000 kleiner Bildschirm	655,50
8507	I/O-TOOL RS-232/2 CBM 4000 großer Bildschirm	655,50
8508	I/O-TOOL RS-232/2 CBM 8000 ASCII	655,50
8509	I/O-TOOL RS-232/2 CBM 8000 DIN	655,50
	Rechnerkopplung zwischen CBM-Rechnern und EPSON HX-20:	
8530	I/O-TOOL HX-20 für CBM 3000	598,50
8531	I/O-TOOL HX-20 für CBM 4000 kleiner Bildschirm	598,50
8532	I/O-TOOL HX-20 für CBM 4000 großer Bildschirm	598,50
8533	I/O-TOOL HX-20 für CBM 8000 eckig oder SK/ASCII	598,50
8534	I/O-TOOL HX-20 für CBM 8000 eckig oder SK/DIN	598,50
	Datenerfassung am EPSON HX-20 als Ansteckeinheit:	
8550	HX-20 I/O-EXPANSION UNIT DIGITAL	934,80
8551	HX-20 I/O-EXPANSION UNIT ANALOG, Basisversion	792,30
8552	HX-20 Centronics Interface zum Anschluß an Drucker	450,30

Preise inkl. MwSt., Kataloge kostenlos.

Informieren Sie sich über unseren einmaligen Service: unsere SOFTWARE-PFLEGEGARANTIE, Kataloge '84 kostenlos. Interessante Informationen bieten wir Ihnen während unserer hauseigenen Ausstellung:
COMPUTER-SHOW IN KOBLENZ – Neuheiten auf 1200 qm. Eintritt kostenlos. Wir freuen uns auf Ihren Besuch!

Beratung · Entwicklung · Fertigung · Betreuung: Vier komplexe Bereiche – e i n Ansprechpartner!

Görlitz Computerbau · Lambertstraße 49 · Postfach 852 · 5400 Koblenz · Telefon: 0261/27500

Inhalt

Grundlagen, Teil 1 ..... Seite 4
Grundlagen, Teil 2 ..... Seite 7
Grundlagen, Teil 3 ..... Seite 9
Grundlagen, Teil 4 ..... Seite 12
Grundlagen, Teil 5 ..... Seite 16
Grundlagen, Teil 6 ..... Seite 18
Grundlagen, Teil 7 ..... Seite 21
Grundlagen, Teil 8 ..... Seite 22
Grundlagen, Teil 9 ..... Seite 25
Grundlagen, Teil 10 ..... Seite 27
Grundlagen, Teil 11 ..... Seite 28
Grundlagen, Teil 12 ..... Seite 29
Grundlagen, Teil 13 ..... Seite 32
Der Standard-Data-Generator ..... Seite 35
Der ComputerSchau-Maschinensprache-Monitor (Com Mon 64) .... Seite 39
Arbeiten mit Diskette ..... Seite 50
Hardware-Tips 1 ..... Seite 60
Hardware-Tips 2 ..... Seite 61
Hardware-Tips 3 ..... Seite 63
Hardware-Tips 4 ..... Seite 65
Hacker – ein Spiel nach der Wirklichkeit ..... Seite 67
Programme ..... Seite 74
Verschiedenes ..... Seite 91

Impressum

1984

Franzis-Verlag GmbH, Karlstraße 37–41, 8000 München 2.
Sonderheft der Zeitschrift ComputerSchau.
Bearbeitet von Lothar Miedel, Redaktion der Zeitschrift ComputerSchau.
Für den Text verantwortlich: Lutz Findeisen.
Titelgestaltung: Alois Mangler.
© Sämtliche Rechte – besonders das Übersetzungsrecht – an Text und Bildern vorbehalten. Fotomechanische Vervielfältigung nur mit Genehmigung des Verlages. Jeder Nachdruck, auch auszugsweise, und jede Wiedergabe der Abbildungen, auch in verändertem Zustand, sind verboten.
ISSN 0177-2139
Druck: Franzis-Druck GmbH · Karlstraße 35, 8000 München 2
ZV-Artikel-Nr. 20106 · F/ZV/1184/1058/15'

Ablage von Basicprogrammen im Speicher des C 64

Der Basicbereich des Commodore 64 beginnt normalerweise ab der dezimalen Adresse 2049 und endet bei 40959; dies entspricht dem sedizimalen (hexadezimalen) Bereich von 0801...9fff. Das heißt, daß üblicherweise im Bereich dieser Adressen ein Basicprogramm abgelegt wird. Wie dies im Computer geschieht, wollen wir uns nun etwas genauer ansehen.

Maschinensprachemonitor erforderlich

Um es nun aber näher untersuchen zu können, benötigen wir einen Maschinensprache-Monitor – also ein Programm, welches es ermöglicht, die Speicherzellen des C64 näher zu betrachten. Denn ohne diesen ist das ganze zu umständlich.
Leider hat der C64, im Gegensatz zu den meisten seiner „Geschwister", keine derartige „Firmware" eingebaut. Deshalb muß auf einen der vielen im Umlauf befindlichen – z.B. auf den in unserer Schwesterzeitschrift „MC" erschienenen – Monitor oder auf den in diesem Heft abgedruckten COMMON 64 zurückgegriffen werden. Mit letzterem sind all die Dinge, die wir auf den nächsten Seiten gemeinsam erarbeiten werden, nachzuvollziehen. Für spätere und ernsthaftere Arbeiten sollte aber möglichst ein „besserer" Maschinensprachemonitor verwendet werden. Nachdem nun ein Monitor geladen und gestartet wurde, begeben wir uns wieder zurück ins Basic und geben ein kleines Programm ein.

Damit alles genau mitverfolgt werden kann, sollte nun auch alles genauso eingegeben werden, wie wir dies nun aufzeigen. Denn bei Abweichungen gibt es keine Übereinstimmung. Dies gilt nun nicht nur für die erste Untersuchung, die Sie und wir gemeinsam durchführen wollen, sondern auch für alle weiteren.

```
10 PRINTCHR$(147)
20 PRINT"COMPUTERSCHAU"CHR$(13)
30 PRINT"ZEIGT,";
40 PRINT"WIE MAN'S MACHT!"

PROGRAMM 1 = BASISPROGRAMM
```

Dieses Programm sollte nach der korrekten Eingabe gleich abgespeichert werden, da es für spätere Untersuchungen und Versuche immer wieder gebraucht wird!

Arbeiten mit dem Monitor

Nach der Eingabe dieser Zeilen wird in das Monitorprogramm gesprungen und der Bereich von $0800...0848 als Hexdump*) aufgerufen. Dadurch erfolgt bei den meisten Maschinensprachemonitoren die Ausgabe bis einschließlich $08ff. Dies liegt daran, daß bei den Ausgaben immer die Zeilen bis zum Ende geschrieben werden.

*) *Hexdump ist die sedezimale Ausgabe eines Speicherbereiches*

Die Bildschirm-Ausgabe des Hexdumps sieht nun, je nach verwendetem „Monitor" etwa so aus, wie Bild 1.

Analyse der Programmablage

Die Speicherstelle $0800 enthält die Hexzahl „00", dies soll uns im Augenblick aber noch nicht weiter interessieren. Die Adressen $0801 und $0802 beinhalten die sogenannten „Linker" (= Verweiszeiger) auf die nächste Basiczeile.

„Linker zeigen den Weg"

In unserem Beispiel sind dies die beiden Hexzahlen 0d und 08. Da diese beiden Bytes in Low-/Highbyte-Konfiguration abgelegt wurden, braucht man diese nur in der Reihenfolge auszutauschen und hat dann bereits die Adresse des Beginnes der nächsten Basiczeile. In unserem Falle also: $080d. Daß dies auch richtig ist, zeigt uns $080d, denn dort befindet sich als Wert „00", was nichts anderes bedeutet,


```
.M 0800,0848
.: 0800 00 0D 08 0A 00 99 C7 28
.: 0808 31 34 37 29 00 27 08 14
.: 0810 00 99 22 43 4F 4D 50 55
.: 0818 54 45 52 53 43 48 41 55
.: 0820 22 C7 28 31 33 29 00 36
.: 0828 08 1E 00 99 22 5A 45 49
.: 0830 47 54 2C 22 3B 00 4E 08
.: 0838 28 00 99 22 57 49 45 20
.: 0840 4D 41 4E 27 53 20 4D 41
.: 0848 43 48 54 21 22 00 00 00

HEXDUMP DES BASISPROGRAMMES
```

Bild 1

als daß die Basiczeile dort zu Ende ist.
Also wieder zurück. In den beiden erstgenannten Bytes stehen also die Linker für die nächste Basiczeile. Anschließend folgen die beiden Bytes für die Basiczeilen-Nummer. Ebenfalls wieder als Low-/Highbyte-Wert. Deshalb lautet also die Zeilennummer $000a (nicht $0a00) und dies ist in Dezimal die Ziffer „10". Übrigens, Sie können den Mini-Hex-Monitor, der sich als Listing in diesem Heft befindet, auch zur Umwandlung von Sedezimal- in Dezimalzahlen benützen. Sie brauchen nur entweder bei der Frage nach der „Startadresse" oder der „Dumpadresse" die entsprechende Hexadezimalzahl einzutippen. In der Ausgabemaske erscheint bei diesem Programm hinter der Hexausgabe auch der dezimale Wert. Doch nun wieder weiter mit der Untersuchung unseres kleinen Programms.

Computer-„Kürzel" = Token

In der nächsten Speicherstelle finden wir die Zahl $99; dies ist das „Token" des Basicbefehles „print". Token's sind die durch ein Byte dargestellten Basic-Befehlswörter, wie print, rem, open, clr, list, sys ... usw.
Nach $99 folgt $c7, ebenfalls ein Token, für den Ausdruck „chr$". Danach ist die hexadezimale Folge: 28,31,34,37,29 und dies alles ist nichts anderes als: „(147)". Anschließend folgt dann „00" und das bedeutet für das Betriebssystem, daß die Basiczeile zu Ende ist. Die Zeile 10 ist lediglich dazu da, den Bildschirm zu löschen. Man hätte dies natürlich auch mit dem entsprechenden Steuerzeichen erreichen können, doch dann wäre im Listing das inverse „Herzchen" erschienen und gerade für Beginner ist die Identifikation derartiger Zeichen nicht immer gerade einfach. Außerdem ist es wirklich ganz wichtig, daß das Programm bei Ihnen vollkommen identisch mit dem unsrigen ist, denn sonst stimmt es ja nicht überein und auch die nachfolgenden Versuche führen nicht zu den gewünschten Erfolgen. Doch nun wieder weiter mit dem Basisprogramm.
Bei $080d beginnt die nächste Basiczeile und hier ist es nun wieder identisch mit der ersten. Es beginnt mit dem Linker zur nächsten Zeile, dann folgt die Zeilennummer, dann der Zeileninhalt.
Also: nächste Basiczeile bei $0827, Zeilennummer $14 (= dezimal 20), Token für „print", Code für Anführungszeichen, danach folgt nun der Basictext „COMPUTERSCHAU". All diese Zeichen nun sind lediglich der ganz normale ASCII*)-Text, wie man leicht nachprüfen kann. ($43 = dezimal 67 – und chr$(67) ist der Buchstabe „c".)

Wie bei der Schreibmaschine

Nach dem Wort COMPUTER-

*) *(ASCII = American Standard Code for Information Interchange)*

Ein Sonderheft der
Elektronik
zu einem Thema, das die Welt verändert hat:

Zweite erweiterte Auflage
Daten-Kommunikation

Alles Wichtige, was die ELEKTRONIK bisher über dieses Spezialgebiet schrieb, finden Sie jetzt komplett in einem Heft.
Damit gibt der Franzis-Verlag Ingenieuren, Informatikern und anderen Interessierten ein vielseitiges Informationsmittel in die Hand, das alle wesentlichen Aspekte der Datenübertragung transparent macht.

Aus dem Inhalt:
Ausdrücke, Methoden, Komponenten-Schnittstellen; Datenübertragungs-Protokolle; Multiplexer; Modems; Lokale Netzwerke; Elektronische Datenvermittlung; Meßverfahren in der Datenkommunikation.
Neu: Breitband oder Basisband; Datenübertragungsdienste der Deutschen Bundespost u. a.

Hier erhalten Sie dieses Sonderheft:

Bei allen Bahnhofsbuchhandlungen, beim Elektronik-Fachhandel, bei größeren Zeitschriftenverkaufsstellen, in Buchhandlungen oder gegen Vorauszahlung direkt vom Franzis-Verlag. Wir bitten Sie in diesem Fall als Bestellung 20,– DM (18,– DM plus 2,– DM Porto) auf unser Postscheckk. München Nr. 813 75-809 mit Angabe „Datenkommunikation" zu überweisen oder einen Scheck über diese Summe einzusenden. Bitte vergessen Sie nicht, auf dem Zahlungsbeleg in Druckschrift Ihre vollständige Anschrift anzugeben. Sofort nach Eingang der Zahlung senden wir Ihnen das Heft zu.

Franzis-Verlag
Karlstraße 37, 8000 München 2
Telefon 0 89/51 17-2 39/-3 80

Sie erhalten dieses Heft in der Schweiz auch beim
Verlag Thali AG
CH-6285 Hitzkirch

und in Österreich beim
Fachbuch Center Erb
Amerlingstraße 1, A-1061 Wien

SCHAU und dem abschließenden Schlußzeichen folgt die Befehlssequenz für „carriage return“, also der Befehl „Wagen-Rücklauf“, wie bei der Schreibmaschine bzw. beim Fernschreiber. Das alles kann natürlich auch mittels entsprechender Steuerzeichen programmiert werden, sehen Sie sich dies ruhig später selbst einmal an, denn Sie wissen nun ja bereits, wie Programme abgelegt werden.
Gehen wir nun gleich zur letzten Basiczeile. Diese beginnt bei $0836. Der Verweiszeiger gibt uns als nächsten Basiczeilenbeginn die Adresse $084e an. Dort steht $00 und in der darauffolgenden Adresse ebenfalls. Das heißt nun, daß das Basicprogramm hier endet.
Zusammenfassend kann also gesagt werden, daß im normalen Basic drei aufeinanderfolgende „00“ immer das Programmende darstellen. (Die erste Null = Zeilenende, die beiden folgenden = Programmende.)
Ja, das war das ganze Geheimnis, wie im Computer C64 ein Basicprogramm abgelegt wird. Sie können ja nun mit eigenen kleinen Programmen weitere Untersuchungen durchführen. Mit dieser Art der Untersuchungstechnik können Sie, wenn Sie wollen, natürlich auch eine Tabelle der „Token's“ Ihres Computers aufstellen; einfacher ist es aber, gleich die fertige Tabelle, die ebenfalls in diesem Heft enthalten ist, anzusehen. Übrigens, bei den anderen, bisherigen Computern der Commodore-Serie ist dies alles nicht anders, nur der Beginn des, bzw. der ganze Basicbereich liegt bei diesen an anderen Adressen. Auch bei Computern anderer Fabrikate erfolgt die Ablage von Basicprogrammen in ähnlicher Art und Weise.

Hokuspokus: Programm da, Programm fort?

So, nun wollen wir einmal unser Basicprogramm „löschen“. Deshalb zurück aus dem Monitorprogramm, „new“ eingeben und die Returntaste drücken. Die Befehlseingabe „list“ zeigt, daß das Programm verschwunden ist. Aber es ist nicht wirklich gelöscht! Sondern ...????
Also wieder ins Monitorprogramm und den gleichen Bereich wie vorher angesehen.
Wie Sie sicher unschwer erkennen können, haben sich nur zwei Speicherstellen in diesem „Dump“ verändert, nämlich $0801 und $0802.

```
.M 0800,0848
.: 0800 00 00 00 0A 00 99 C7 28
.: 0808 31 34 37 29 00 27 08 14
.: 0810 00 99 22 43 4F 4D 50 55
.: 0818 54 45 52 53 43 48 41 55
.: 0820 22 C7 28 31 33 29 00 36
.: 0828 08 1E 00 99 22 5A 45 49
.: 0830 47 54 2C 22 3B 00 4E 08
.: 0838 28 00 99 22 57 49 45 20
.: 0840 4D 41 4E 27 53 20 4D 41
.: 0848 43 48 54 21 22 00 00 00

HEXDUMP NACH DEM BEFEHL "NEW" (RETURN)
```

Bild 2

Dort steht nun anstelle der Linkeradresse $080d das bereits bekannte „Basicende“, und nur dies ist der Grund, weshalb das Programm verschwunden ist. Den Beweis dafür treten wir gleich gemeinsam an. Schreiben Sie bitte in die beiden Speicherzellen wieder die Werte, die ursprünglich dort waren.

Programm da

So, nun wieder aufgelistet, das Programm ist wieder zu sehen. Durch den Befehl „new“ wird also das Programm nicht wirklich gelöscht, sondern lediglich im Basictext zwei Speicherstellen verändert. Allerdings werden auch noch andere Speicherstellen verändert. Deswegen „new“ (return) eingeben, den Cursor auf die Zeilennummer zehn stellen und viermal die Returntaste drücken. Damit ist das Programm wieder korrekt im Computer und auch die verschiedenen, rechnerinternen Verweiszeiger für Beginn der Variablen usw. sind wieder im ordnungsgemäßen Zustand.
Programmveränderungen mittels Monitor betreffen natürlich nicht nur die „Basic-Linker“, sie können alle Bytes des Programmes verändern!
Erwarten Sie aber nicht, daß sich diese Veränderungen immer „positiv“ auf Ihr Programm auswirken. Abgesehen davon, daß Sie damit ein im Speicher befindliches Basicprogramm total zerstören können, kann es auch passieren, daß Sie durch versehentliche Manipulationen im RAM-Bereich das Betriebssystem des C64 derart stören, daß nur noch ein „Netzschalter-Reset“ weiterhilft. Sind Sie bitte deshalb besonders vorsichtig bei Veränderungen, welche außerhalb Ihres Basicprogrammes liegen, vor allem im Bereich unterhalb von $0400.
Versuchen Sie aber trotzdem ruhig einmal alle möglichen Speicherplätze zu verändern, denn: „Übung macht den Meister“.
Wir hoffen, es hat Ihnen etwas Spaß gemacht, Ihrem C64 etwas näher ins Programm zu schauen.
Haben Sie aber vergessen, das „new“ einzugeben, dann hat sich der C64 „aufgehängt“, da hilft nur noch ab- und wiedereinschalten.

Programm fort

So, nun lassen wir unser Programm wieder verschwinden. Wie das geht? Ganz einfach, Sie wissen es bereits mit „new“. Doch bestimmt haben Sie auch die andere Methode „erkannt“, ...richtig die bereits bekannten Linkadressen mit Nullen aufgefüllt. Zurück ins Basic, Listversuch ... weg ist es.
So, im nächsten Abschnitt wollen wir uns noch etwas weiter im Speicher umsehen.

Vorgang beim „Listen“ eines Basicprogrammes

Der Befehl „List“ (return) bewirkt, daß das Betriebssystem des Computers, beginnend bei „Basicstart“ in etwa ebenso vorgeht, wie wir dies bei der Untersuchung unseres kleinen Programmes getan haben. Zuerst wird „nachgesehen“ wo eigentlich Basicstart liegt. Diese Information ist in 2 speziellen Speicherstellen ($2b und $2c, dies sind in Dezimal ausgedrückt die Adressen 43 und 44) abgelegt. Die Eingabe folgender Befehlssequenz zeigt uns also, wo „Basicbeginn“ ist:

printpeek(43)+peek(44)*256
Im Normalfalle muß der Computer mit der Ausgabe „2049“ antworten. Durch „list“ beginnt die Abarbeitung einer Betriebssystemroutine, welche aus den Speicherstellen 43 und 44 die Anfangsadresse holt.
Nach verschiedenen Überprüfungen durch diese Routine (Abfrage ob nur List, ob List bis Zeilennummer, ob List Zeilennummer bis Zeilennummer usw.) beginnt dann die Ausgabe des Programmtextes. Stößt die Listroutine dann auf die uns schon bekannte „00“, dann wird die Ausgabe dieser Zeile abgebrochen und mit der nächsten weitergemacht, bis zum Programmende.

Listvorgang und Programmablauf sind unterschiedlich!

Diese Eigenart wird sehr gerne dazu benutzt, um verschiedene Listschutzarten auf- bzw. in Programme einzubauen. Wie wir oben bereits mehrmals erwähnt haben, erkennt das Betriebssystem durch eine „00“ das Zeilenende und „hangelt“ sich über die Linker durch das Programm. Dies wollen wir uns gleich einmal zunutze machen und mit mehreren Beispielen die verschiedensten Effekte und Auswirkungen aufzeigen.
Ausgangsbasis für die also nun folgenden Versuche ist unser „Basisprogramm“, mit welchem wir bereits die „Ablage im Speicher“ untersucht haben.
Falls dies noch unverändert im C64 steht, können wir bereits sofort weitermachen, ansonsten muß es leider wieder geladen werden. Sie hatten es doch hoffentlich wie eingangs gebeten, gleich abgespeichert?
Geben Sie nun bitte folgendes ein:
Poke 2049,142 (return)
Für diejenigen, die noch im „Monitorbetrieb“ sind: bitte die Speicherstelle $0801 (alter Wert: $0d) abändern in $8e und dann den Monitor verlassen, also zurück zum normalen Basic!
Der Befehl „List“ zeigt uns nun lediglich noch die Basiczeile 10 und deren Inhalt. Die Zeilen 20, 30 und 40 werden nicht mehr ausgegeben. Der Befehl „Run“ aber macht deutlich, daß diese nicht verlorengegangen sind, sondern lediglich bei der Listausgabe unterdrückt wurden. Das heißt, unsere Manipulation beeinflußte lediglich den Listvorgang, aber nicht das „Abarbeiten“ des Programms. Wird nun entweder durch „Poken“ oder unter Zuhilfenahme des Monitors dieser Wert wieder korrigiert, erscheint das Programm wieder korrekt beim „listen“.
So, nun verstellen wir den Zeiger einmal „irgendwohin“. Wir geben ein:
poke2049,20
Im Monitorbetrieb muß die Zahl $14 eingegeben werden.
Sehen Sie bitte nun selbst auf ihren Bildschirm, was als „Listing“ ausgegeben wird! Durch den Befehl „run“ aber wird das Programm immer noch korrekt ausgeführt.
Den ganzen „Scherz“ können wir nun auch noch anders machen. Anstelle des Wertes $14 bzw. dezimal 20 beim „poken“ können wir nun auch einmal den Wert $27 bzw. dezimal 39 einsetzen. Daraufhin werden die Zeilen 10, 30 und 40 ausgegeben, die Zeile Nr. 20 fehlt nun beim Listvorgang, aber bei „run“ ist alles beim alten. Eine Rückstellung auf die Originalwerte ist natürlich ebenso möglich wie vorher.

Auch die erste Zeile verschwindet

Wir haben nun aufgezeigt, wie man ganze Zeilen verschwinden lassen kann und wollen nun noch weitergehen, denn die erste Zeile blieb ja bisher in allen Fällen immer noch „original“ erhalten. Bitte stellen Sie nun den „Urzustand“ des Programmes wieder her und ergänzen bzw. verändern die Zeile Nummer 10 folgendermaßen:

:REM::::::::::::::::::::
(nach dem REM folgen 21 Doppelpunkte).

Das Programm sollte nun korrekterweise so aussehen:

```
10 PRINTCHR$(147):REM::::::::::::::::::::
20 PRINT"COMPUTERSCHAU"CHR$(13)
30 PRINT"ZEIGT,";
40 PRINT"WIE MAN'S MACHT!"
READY.
```

Nun wieder das „Ganze“ mittels Monitor näher angesehen. Es ist prinzipiell alles beim „alten“ geblieben, aber es hat sich durch diese Ergänzung nun natürlich einiges im Speicher verschoben. Wer Lust hat, kann dies gerne überprüfen.

```
M 0800,0860
:0800 00 24 08 0A 00 99 C7 28 [illegible]
:0808 31 34 37 29 3A 8F 3A 3A 147):■::
:0810 3A 3A 3A 3A 3A 3A 3A 3A ::::::::
:0818 3A 3A 3A 3A 3A 3A 3A 3A ::::::::
:0820 3A 3A 3A 00 3E 08 14 00 [illegible]
:0828 99 22 43 4F 4D 50 55 54 ■"COMPUT
:0830 45 52 53 43 48 41 55 22 ERSCHAU"
:0838 C7 28 31 33 29 00 4D 08 [illegible]
:0840 1E 00 99 22 5A 45 49 47 [illegible]"ZEIG
:0848 54 2C 22 3B 00 65 08 28 T,";[illegible]
:0850 00 99 22 57 49 45 20 4D [illegible]"WIE M
:0858 41 4E 27 53 20 4D 41 43 AN'S MAC
:0860 48 54 21 22 00 00 00 00 HT!"[illegible]
HEXDUMP MIT ASCII-AUSGABE
```

Für unsere weiteren Untersuchungen aber interessieren uns augenblicklich nur die Speicherzellen im Bereich von $080e...$0822. Diese beinhalten im Moment alle den Wert $3a und sollen nun in $14 abgeändert werden. Wenn dies durchgeführt ist, dann bitte wieder auflisten! Ja richtig –, die Zeile zehn ist verschwunden!! Ein schnelles Auge aber sieht, daß sich die verschwundene Zeile kurzzeitig auf dem Schirm gezeigt hatte. Macht aber nichts, denn es muß ja nicht immer die erste Zeile eines Programmes sein, die man unsichtbar machen will!
Bei einem langen Programm aber würde auch ein „schnelles Auge“ bald ermüden, wenn es auf solche kurzzeitigen „Auftritte“ von Zeilen achten soll.

Kein Programm – läuft doch!

Und nun wollen wir noch das ganze Programm verschwinden lassen. Deshalb nun bitte die Speicherstelle $0801 abändern in $65 – fertig.
So, nun ist das ganze Programm verschwunden. Allerdings, wie bereits erwähnt ist die erste Zeile kurz sichtbar. Nun speichern wir wieder einmal sicherheitshalber das Programm (unter einem anderen Programmnamen) ab. Nach einem erneuten Einlesen und nachfolgendem Auflisten, sind plötzlich die Zeilen 20, 30 und 40 wieder sichtbar! Der erste Linker wurde also wieder korrigiert! Durch Poke2049,101 sind die Zeilen wieder (fast) verschwunden. Ein angeschlossener Drucker aber, der über CMD/list (der Druckerkanal muß natürlich richtig „geöffnet“ werden) um Programmlistingausgabe gebeten wird, gibt die Zeile 10 bis zum REM aus.
Doch damit nun mal Schluß mit dieser Technik.
Es gibt noch weitere Möglichkeiten, Listschutzarten zu realisieren. So z.B. die Methode nach einem REM und nachfolgenden „geshifteten“ Buchstaben L bei der Eingabe noch weitere Zeichen (z.B. Copyrightvermerk) einzutippen und diese oder einen Teil davon später durch eine kleine Maschinenroutine, die überprüft, ob das „Copyright“ noch original bzw. überhaupt vorhanden ist, zu ersetzen usw.

Ausnutzung Betriebssystemfehler

Interessant bei dieser Art ist, daß der Listvorgang durch einen Fehler im Betriebssystem, nämlich durch das „geshiftete L“ abgebrochen und „? Syntax error“ ausgegeben wird. Listet man aber erst nach dieser Zeilennummer, dann kommt das Programm ab dieser wieder ganz normal zur Ausgabe bis zum nächsten evtl. eingebauten Scherz. So, nun das letzte Wort des letzten Satzes beiseite. Die ganzen Listschutzmethoden beruhen alle auf diesen bzw. ähnlichen Prinzipien! Egal, wie die Eingabe dieser Methoden erfolgt, im Endeffekt wird nur erreicht, daß die Bildschirmausgabe beeinflußt wird. Wir wollten nicht zeigen, wie Programme „geknackt“ werden, sondern mehr Verständnis vermitteln, wie Computer bzw. Interpreter arbeiten. Diejenigen, die wirklich „knacken“ wollen, kennen diese und auch noch weit bessere Methoden für den Programmschutz und auch entsprechende Gegenmaßnahmen. In vielen Publikationen sind im Laufe der letzten Jahre die verschiedensten Methoden für Listschutztechniken beschrieben worden. Und in manchen Fällen war der Urheber bestimmt der Meinung, diesen Schutz kann niemand entfernen oder entdecken. Oftmals wurden verschiedene Methoden miteinander verknüpft, es wurden Codes abgefragt usw., aber ist es denn noch ein Schutz, wenn dieser veröffentlicht wurde? Oder haben die Entwickler derartiger Methoden keinen Maschinensprachemonitor?

Es gibt keinen „echten“ Listschutz

Deswegen möchten wir alle diejenigen vor der wirklich unvernünftigen Zeitinvestition zur Entwicklung von Listschutztechniken warnen, die glauben, daß derartige Manipulationen eines Programmes nicht erkannt werden können. Es ist bestimmt nur eine sehr, sehr kurzfristige Verhinderung des Einblickes in ein Programm. Wem es natürlich Spaß macht, mit seinem Freund oder Bekannten „listgeschützte“ Programme auszutauschen und dies als eine Art Sport oder Wettbewerb sieht, um festzustellen, wie lange jemand braucht, um hinter den kleinen „Dreh“ zu kommen, soll sich bitte nicht durch unsere letzten Ausführungen davon abhalten lassen. Denn auch dadurch lernt man seine „Maschine“ besser kennen und verstehen.

Verschieben des Basicbeginns

Im Abschnitt über den Vorgang beim Listen haben wir geschrieben, daß wir nochmals auf den „Basicbeginn“ zurückkommen wollten! Zur Erinnerung – der Interpreter holt aus den Speicherzellen $2b (43) und $2c (44) die Information, wo sich der Basicbeginn befindet.

Durch Veränderung der beiden Speicherinhalte kann der Basicbeginn an eine andere Adresse gelegt werden. Wie das geht, erfahren Sie nun.
Nachdem wir in den anderen Abschnitten bereits mit dem Monitor gearbeitet haben, dürfte es nun kein größeres Problem sein, mit diesem auch weiter zu arbeiten. Allerdings wird er für einen Teil unserer Ausführungen nicht unbedingt erforderlich sein, denn die Auswirkung unserer Veränderungen können wir auch ohne ihn feststellen. Aber wir wollen doch ganz genau wissen, was sich im Computer tut, oder?

Wo ist Basicbeginn?

Zunächst einmal fragen wir durch „Peeken“ die Inhalte beider Speicherzellen ab. Also:

```
printpeek(43),peek(44)
```

Der C64 müßte nun mit der Ausgabe der Werte eins und acht antworten.

Um es nochmals nachzurechnen, Basicbeginn ist bei 8 × 256 + 1 = 2049.

Dies ist also korrekt. Nun haben wir in den vorhergehenden Abschnitten gesehen, daß es sehr einfach war, Werte im Programmspeicherbereich zu verändern. Das gleiche gilt natürlich auch für alle anderen RAM-Bereiche. Deswegen also den C64 in den „Einschaltzustand“ gebracht. Dies deshalb, damit die Ausgangsbedingungen klar und nicht durch eventuelle vorausgehende Manipulationen Unterschiede vorhanden sind. So und nun gleich ans Werk: „Frisch gepoked ist halb gewonnen“.

```
poke44,9:new (return)
```

oder für Monitorliebhaber (irgendwann wird jeder Freak einer): Änderung des Inhaltes der Speicherstelle $2c in den Wert $1a, raus aus dem Monitor und „new“ (return) eingeben.

0 = nichts?

Aber halt! Wer nun zu schnell war, hat einen großen Fehler gemacht. Denn nun versteht der C64 plötzlich sein Basic nicht mehr. Er quittiert das „new“ mit einem „Syntax error“. Eines muß nämlich noch vorher geschehen: in die Speicherstelle vor dem eigentlichen neuen Basicbeginn muß die Zahl Null eingeschrieben werden. Erst dann arbeitet unser Betriebssytem wieder korrekt. Also:

```
poke44,9:pokepeek(43)+
peek(44)*256-1,0:new
```

Nun erfolgt keine Fehlermeldung. Aber was ist nun geschehen? Wie sich bestimmt jeder Leser erinnert, meldet sich der C64 nach dem Einschalten unter anderem mit der Information: „38911 Basic bytes free“. Bei der Abfrage durch den Befehl: printfre(0) antwortet er mit –26883.

Zwischendurch bemerkt

Ein Fehler im Betriebssystem ist für diesen Effekt verantwortlich. Die echte Anzahl der freien Speicherstellen erhält man durch folgende Eingabe:

```
printfre(0)+65535 (return) nämlich:
38908
```

Dieser Wert stimmt aber auch nicht mit dem nach dem Einschalten ausgegebenen überein! Dies soll uns aber nicht weiter stören, denn dies liegt daran, daß wir einen Befehl eingegeben haben.

Es geht weiter

Also, normalerweise müßte bei leerem Basicprogramm-Speicher und der Frage nach der Anzahl der freien Bytes als Antwort „–28627“ kommen. Geben wir aber nun nach der weiter oben durchgeführten Manipulation die Frage nach dem freien Speicherplatz an unseren Computer, so antwortet dieser nun mit: „–26883“!
Oder nach der „Additionsmethode“ gefragt: mit „38652“. Das heißt, wir haben nun 38908 minus 38652 = 256 Bytes weniger frei. Ganz genau gesagt, haben wir unseren Basicstart um 1*256 Bytes nach oben verschoben, das heißt: er liegt nun nicht mehr bei $0801 sondern bei $0901 bzw. bei dezimal 2305.

Doch wozu nun dieses alles?

Es wurde dadurch doch augenscheinlich nur etwas „negatives“ durchgeführt, wir haben unseren Basicbereich um ¼ KByte „beraubt“. Aber sehen wir uns das gan-

ze nun erst wieder etwas genauer an. Also wieder mit dem Monitor arbeiten. Aber vorher bitte erst wieder das Basisprogramm laden. Bitte aber nicht absolut, sondern verschieblich, das heißt mit der Befehlsfolge:

Load"programmname",8 oder Load-"programmname",1

(Je nachdem, ob mit Floppy oder Kassette gearbeitet wird.) Wichtig dabei ist, daß nicht mittels des Monitorladebefehl geladen wird!
Der Befehl „run" zeigt, daß es einwandfrei abläuft. Nun mit dem Maschinensprache-Monitor wieder den Bereich $0800...$0839 ausgeben lassen. Sie werden nun feststellen, daß das ursprünglich in diesem Speicherbereich befindliche Programm dort nicht mehr vorhanden ist. Ein Hexdump des Bereiches $0900...$0948 aber zeigt, daß das Programm nun dort abgelegt ist.
Wir haben also wirklich unseren Basicbereich, von unten her, eingeengt. Aber dies hatte schon einen ganz bestimmten Grund, und es kann einige hierfür geben.
Einen möchten wir nun aufzeigen. Die Programmiersprache Basic ist nämlich für verschiedene Anwendungen zu langsam. Deswegen werden entsprechende Routinen, bei denen es auf Geschwindigkeit ankommt, in Maschinensprache geschrieben.

Platz für Maschinenprogramme

Kleinere Maschinenroutinen legt man meist im Kassettenrecorderpuffer ab. Wenn aber dieser Platz nicht reicht, können größere „Maschinensprachteile" auch beispielsweise im Bereich $c000...$cfff abgelegt werden. Auch im oberen Basicbereich kann dies erfolgen, wenn man dem C64 mitteilt, daß er weniger Speicherplatz hat. Dazu müßte dem Computer dies durch „poken" in die entsprechenden „Informationsspeicherstellen" mitgeteilt werden. Sie sehen, es gibt verschiedene Methoden, den Speicherbereich einzuengen.
Ebenso gibt es verschiedene Arten, Maschinenprogrammroutinen im normalen RAM-Bereich unterzubringen. Eine Methode ist beispielsweise, diese Programmteile als Datastatements – im Basicprogramm –, über eine for/next Schleife mit read und poke in den gewünschten Speicherbereich zu bringen. Dadurch geht aber relativ viel Speicherplatz verloren. Denn diese Werte sind dann ja, sowohl an der richtigen Speicheradresse, als auch im Basicprogramm zu finden (zuzüglich der Zeilennummer, der for-next-Schleife und der Tokens für Data). Auch dies könnte man durch „Verbiegen der entsprechenden Zeiger" wieder ändern, wenn man die ganzen Datastatements am Ende des Programmes ablegt. Eine andere Methode ist, die Maschinensprachteile zu laden und danach den Basicteil oder

```
M 0827
:0827 11 11 43 4F 4D 50 55 54
:082F 45 52 53 43 48 41 55 20
:0837 4B 4F 4D 4D 54 20 56 4F
:083F 4D 20 46 52 41 4E 5A 49
:0847 53 20 56 45 52 4C 41 47
:084F 0D 00 A0 00 B9 27 08 20
:0857 D2 FF C8 C0 29 D0 F5 60

HEXDUMP MASCHINENPROGRAMM
```

auch umgekehrt. Ob dies nun durch direkte Befehle, oder über einen „Starter" – also ein Programm, das mehrere Teile nachlädt – geschieht, ist egal. Dies aber ist nicht nur umständlich, sondern man hat das „Arbeitsprogramm" auch noch in mehreren Teilen auf seinem Programm-Speichermedium. Bei einer Bereinigung der Programmsammlung kann es dann sehr leicht geschehen, daß man aus Versehen mal ein Programmteil löscht, weil man sich nicht mehr rechtzeitig daran erinnert, daß dies zu einem anderen Teilprogramm gehört. Besser ist es, das Programm komplett zu haben.

Basic- und Maschinenspracheteil in einem Stück

Dazu dient die Methode, die wir eingangs dieses Abschnittes angewandt haben.
Wir können nun nämlich unser Basicprogramm ganz normal schreiben, abspeichern, nach Manipulation des Basicbeginnes wieder einladen, das Maschinenprogramm (für unser Beispiel im Bereich zwischen $0801 und $0900) ablegen, die Speicherstellen 43 und 44 wieder richtig stellen und insgesamt wieder abspeichern.
Das Ganze war nun natürlich etwas sehr kurz und oberflächlich erklärt und dadurch können eventuell Fehler auftreten, die nicht so ganz ohne negative Auswirkungen auf den korrekten Programmablauf dieses Hilfsprogramms sein können.

Praktisches Beispiel

Sinnvollerweise beginnen wir wieder mit dem „Urzustand" des Computers, d.h. wir überprüfen bzw. korrigieren ggf. die Speicherstelle dezimal 44. Dort muß 8 stehen. Durch „new" haben wir erreicht, daß nicht noch irgendetwas, das bei unseren Veränderungen eingeschrieben wurde, irgendwelchen Ärger macht. Nun geben wir eine Veränderungszeile ein.

```
VERAENDERUNGSZEILE

10 POKE44,9:POKEPEEK(43)+PEEK(44)*256-1,0:RUN
```


Nun „run" und passiert ist es. Nachdem wir nun dem C64 mitgeteilt haben, daß sein Basicbeginn bei $0900 ist, laden wir also unser Basisprogramm (Programm 1) relativ.
Daß es sich ganz normal auflisten läßt und auch „läuft", möchten wir schon fast nicht mehr erwähnen.
Nun kurz gepokt: poke 44,8, danach „list". Es ist jetzt wieder die Veränderungszeile zu sehen. Poke 44,9 – und das Basicprogramm ist wieder da.
Aber wir wollten ja ein Maschinenprogramm zwischen die beiden Basicteile schreiben. Deswegen nun mittels des Monitors die Speicherstellen $0827...$085e beschrieben, wie dies das Hexdumplisting des Maschinenprogrammes zeigt.
Aus dem Monitorbetrieb heraus und das Basicprogramm durch die Zeilen 50 und 60 ergänzt. Siehe „Erweitertes Basisprogramm".
Danach poke 44,8... save"Komplettprogramm",8 (oder 1) ... fertig!
Nun einfach „run" eingeben, den Erfolg oder Mißerfolg können Sie dann sofort auf dem Bildschirm sehen.
Wenn alles korrekt ist, können Sie das komplette Programm nun jederzeit in einem Zuge laden und ablaufen lassen. Daß der Basicbeginn natürlich nicht nur nach $0900 gelegt werden kann, hätten wir fast vergessen, zu erwähnen.
Hoffentlich hat es Ihnen Spaß gemacht!

```
10 PRINTCHR$(147)
20 PRINT'COMPUTERSCHAU'CHR$(13)
30 PRINT'ZEIGT,';
40 PRINT'WIE MAN'S MACHT!'
50 SYS2129
60 POKE44,8

ERWEITERTES BASISPROGRAMM
```

PRINT TECHNIK
IHR PARTNER FÜR 1A BAUTEILE/LEITERPLATTEN/COMPUTER
AUTORISIERTER COMMODORE-GROSSHÄNDLER FÜR VC 20 + VC 64
SPEZIALIST FÜR COMMODORE PERIPHERIEENTWICKLUNGEN
1060 WIEN, Stumpergasse 34 / Tel. (0043-222) 57 34 23 / Telex 112 996

Interface für Commodore C 64

SPEICHEROSZILLOSKOP
Mit unserer neuesten Bausatzentwicklung wird es nun endlich möglich, extrem langsame wie auch schnelle Abläufe preiswert zu speichern und OSZILLOGRAPHISCH darzustellen. Steckmodul mit AD-Wandler, Hardware-Triggerung, Y-Verstärkung (10 V, 1 V, 100 mV) pro Rastereinheit, dazugehörige komfortable Software mit kalibrierter Zeitbasis und Raster. Darstellung auf 1–95 Bildseiten möglich – ausplotten auf VC 1520 möglich.
Fertiggerät DM 398.–
Bausatz DM 298.–

VIDEO-DIGITIZER
Ein neuer PRINT-TECHNIK-Bausatz, mit dem man jedes Videosignal in ca. 4 sec. in den Speicher eines Commodore VC 64 einlesen kann. Ein Grafikausdruck, Abspeichern auf Floppy sowie Weiterverarbeiten der Bilder ist möglich. Inkl. Software. Farbausdruck möglich mit CANON A 1210
Fertiggerät DM 598.–
Bausatz DM 458.–

UNIVERSAL MODEM
NEUENTWICKLUNG!!!
Universal Modem (Bausatz) für sämtliche Normen, Baudraten (auch BTX-Normen), das die Kommunikation zwischen Computern, bzw. Computer mit einer Datenbank, usw., über Leitung und Telefonnetz ermöglicht.
Bausatz DM 348,–

VOICE MASTER
NEUHEIT!!!
Der Voice Master, Hard- + Software, ist ein einzigartiges Instrument, welches erlaubt Worte und Sätze digitalisiert zu speichern und in beliebiger Reihenfolge sogar in Ihrem eigenen Programm wieder abzuspielen.
Fertiggerät DM 348,–

Interface für Commodore C 64

MICROTRON COMPUTERPRODUKTE
SCHWEIZ
2542 PIETERLEIN
BAHNHOFSTR. 2
TEL. 032/87 24 29

COMPUTER PERIPHERIEN
Pitt · Joern Brockner
BRD/EWG
8000 MÜNCHEN 40
HEIDELBERGERSTR. 6
TEL. 089/36 81 97

GEGEN EINSENDUNG VON ÖS 35.–/SF 5.–/DM 5.– ERHALTEN SIE UNSEREN KATALOG ZUGESANDT.
TÄGLICHER NACHNAHMEVERSAND – WIR SIND FÜR SIE DA – MONTAG–FREITAG 9–18 UHR

Es wird schwieriger

Im letzten Abschnitt war zu lesen, wie der Basicprogrammspeicherbeginn (auf neudeutsch: Start of Basictext) verändert werden kann. Nun wollen wir aufzeigen, wie dies am anderen Ende (ebenfalls auf neudeutsch:) „Highest adress used by basic“ geschieht.
Die Information für den Rechner steht in den Speicherstellen 55 und 56 ($37,$38). In $37 „00“ und in $38 „a0“. Das heißt, der oberste RAM-Speicherplatz, der normalerweise von Basic aus benutzt werden kann, ist:

10·4096–1 = 40959

Damit Sie aber mit dem Computer fit werden, stiften wir nun erst einmal etwas Verwirrung.
Die Commodore C64 haben als Herz des Ganzen einen Mikroprozessor Typ 6510A. Dieser gehört zur Familie der 65xx-Serie und zählt zu den „8-Bitern“.
Im Klartext bedeutet dies, daß der Datenbus 8 Bit breit ist, der Adreßbus aber 16. In den Schaltplanunterlagen werden diese Anschlüsse meist mit D0...D7 (Datenleitungen) und A0...A15 gekennzeichnet. Daraus folgt nun, daß die maximale Zahl, die auf den Datenleitungen dargestellt werden kann, 2 hoch 8 minus 1, also 255 ist. Die Null stellt auch einen Wert dar, deshalb sind also 256 Kombinationen möglich. Der Prozessor kann mit seinen Adreßleitungen insgesamt 2 hoch 16, also 65536 Speicherstellen ansprechen. Wie dies genau geschieht, würde hier zu weit führen. Aber wir wollen nun ein wenig rechnen, dazu können wir natürlich den C64 benutzen, er ist ja ein Rechner.

Eingabe: print 2 ↑ 16
Ausgabe: 65536
minus 1: 65535 = $ffff

Eingabe: print 65536/16
Ausgabe: 4096
minus 1: 4095 = $0fff

Eingabe: print 4096/16
Ausgabe: 256
minus 1: 255 = $00ff

Mit diesen Zahlen hat es eine besondere Bewandtnis, deswegen sollte sich jeder diese, und auch die entsprechende Sedezimalzahl dazu, gut einprägen.

Wollen Sie nur spielen?

Warum nun diese ganze Rechnerei, werden Sie fragen. Nun, einfach deshalb, damit Sie Ihren Computer besser bedienen können. Falls Sie jedoch nur „Pac-man“ (eingetragenes Warenzeichen) oder ähnliches spielen wollen, dann sollten Sie sofort die Zahlen wieder vergessen. Dazu braucht man sie nicht. Vielleicht lesen Sie aber trotzdem weiter.

Pages

Acht Bit bedeuten also einen maximalen Wert von 255, dies sind aber, wir wiederholen es nochmals, 256 Möglichkeiten. Der Computerspeicherbereich ist in sogenannte „pages“ (eingedeutscht „Seiten“) aufgeteilt. Sie ahnen es vermutlich schon, jede „page“ besteht aus 256 Speicherstellen. Der Computer kann 256 pages ansprechen, und 256·256 = 65536.
Sie sehen, die Zahlen tauchen immer wieder auf.
Nachdem der Computer also auch die Ziffer 0 als Wert akzeptiert (anders als wir Menschen teilweise denken: Null = nichts und wo nichts ist, kann auch nichts werden), können Sie sich gedanklich nun den gesamten Adreßbereich von 0...65535 in insgesamt 256 pages mit je 256 Speicherplätzen einteilen.
Für den Computer gibt es also page 0, page 1, ... usw.
Die unteren pages spielen bei unserem Computer eine besondere Rolle. Fangen wir aber der Reihe nach an.

Die unteren Seiten

Wir haben zwar bisher noch nicht davon gesprochen, wo das Betriebssystem im C64 residiert, aber das folgt auch noch. Zuerst einmal können Sie sich ja gut vorstellen, daß jeder Computer, der etwas frei Programmierbares tun soll, auch irgendwelche Plätze braucht für Werte, die er berechnet usw. In viel höherem Maße ist aber für ihn wichtig, bestimmte Werte, die für ihn Basicbeginn, Basicende, Start der Variablen, Cursorfarbe, augenblickliche Cursorposition usw. erkennbar machen, zu finden. Diese Werte werden beim C64 zum größten Teil in diesen unteren pages abgelegt. Irrtümlicherweise hat sich, gerade unter den Hobbyisten, dafür die Bezeichnung „Zeropage“ eingebürgert. In Wirklichkeit aber ist die Zeropage nichts anderes als die page null. Beim C64 haben die unteren 8 Seiten mit den normalen Basicprogrammen nur indirekt zu tun. Diese werden vom Computer selbst verwaltet. Die Pages 4...7 sind normalerweise durch den Bildschirmspeicher belegt. Wieder einmal, diesmal auch wieder zwei kleine, Rechenaufgaben:

4·256 = 1024; 40·25 = 1000

Der Grund für diese Rechnung? Ganz einfach, die Bildschirmausgabe des C64 hat 25 Zeilen, jede mit 40 Zeichen. Da sind doch 24 Bytes über! Richtig, der Bildschirm benötigt nur den unteren Bereich ab 1024, also bis 2023.

Machen Sie doch bitte wieder einmal mit:

```
poke1024,1
:poke2023,5:poke53281,1
```


Sie sehen nun im linken oberen Bildschirmeck ein A (Anfang) und im rechten unteren ein E (Ende). Der letzte Pokebefehl war nötig, da sonst die Zeichen, mit der Farbe des Hintergrundes dargestellt, nicht sichtbar gewesen wären.
Sehen Sie doch mal nach, ob denn überhaupt die Speicherzellen 2024...2047 vorhanden sind. Also:

```
fori=2024to2047:printpeek(i);:next
```

Wie Sie sehen, es werden Werte ausgegeben. Nun sollten Sie versuchen, dort auch einzuschreiben.

```
fori=2024to2047:pokei,
i-2024:nexti
```

und wieder lesen:

```
fori=2024to2047:printpeek(i);:next
```

Die Ausgabe ist dann die Serie der Ziffern 0..23. Also, der Speicher ist da und läßt sich auch „beschreiben“.
Wer also schnell mal ein paar Bytes irgendwo verstecken will, dort ist etwas Platz dafür. Man kann aber auch ohne weiteres den Basicanfang nach unten „schieben“ und noch ein paar Bytes gewinnen. Wie dies geht, können Sie sich ja aus dem vorigen Abschnitt „erlesen“. Das Prinzip ist genauso, vergessen Sie aber bitte nicht, die wichtige Null in die Stelle vor Basicbeginn zu schreiben. Das Verschieben des Basicanfanges in den eigentlichen Bildspeicherbereich hinein ist natürlich unsinnig.
(Anmerkung: Beim Arbeiten mit Sprites werden die 24 Bytes ggf. benötigt!)

RAM-Begrenzung von oben

So, nun wieder zurück zum Ausgangspunkt dieses Abschnittes. Wir wollten Ihnen eigentlich doch zeigen, wie der obere Bereich begrenzt wird. Haben wir ganz schön spannend gemacht, für das, was nun folgt. Aber ehrlich, war nicht auch das „Dazwischen“ aufschlußreich? Doch bevor wir dies zeigen können, müssen wir erst einmal feststellen, wo diese Obergrenze eigentlich normalerweise liegt. Deswegen:

printpeek(55)+peek(56)·256
Der C64 meldet sich mit: 40960

Dies ist die erste Speicherstelle, die üblicherweise nicht von Basic aus benutzt wird. Deswegen also eins abziehen, das ergibt 40959. Erinnern Sie sich? Diese Zahl wurde bereits im Abschnitt über die Ablage von Programmen im Speicher, gleich zu Beginn, genannt.
Außerdem wissen wir, daß der Basicbereich ab 2048 beginnt, deswegen meldet sich der C64 auch beim Einschalten mit 38911 freien Bytes (40959 – 2048 = ?).
Dieses Verständnis war nun wichtig, um die richtigen Zahlen in den Computer einzupoken, wenn der Speicherbereich eingeengt werden soll.

Zum Üben und Training nun ein kleines Programm (siehe unten).

Die Variablen HB und LB sind die Abkürzungen von High- und Lowbyte und diese Werte sind es, die in die Speicherstellen 56 und 55 gepoked werden müssen, um den gewünschten Effekt zu erreichen. Bitte aber nach dem „poken“ den Befehl „new“ oder „clr“ eingeben, erst dann ist alles korrekt durchgeführt und auch erst dann wird die Frage nach den freien Bytes durch den Computer richtig beantwortet.
Nach all dem verstehen Sie wahrscheinlich auch, was mit „High/Low-Byte-Konfiguration“ gemeint ist, oder? Wenn nicht, dann noch eine kleine Hilfe. Erinnern Sie sich noch an die pages? Ja, ... gut! Das Highbyte gibt die Seite und das Lowbyte den Buchstaben auf dieser Seite an. Wenn Sie nun mal nachsehen, was die Speicherplätze 55 und 56 normalerweise beinhalten, dann stellen Sie fest, in 55 steht 0 und in 56 steht 160. Also: Seite 160 der „nullte“ Buchstabe, und das ist:

160·256 = 40960.

So unwichtig war unser „Zwischenausflug“ zu den „pages“ anscheinend gar nicht!
Weitere Beeinflussungen Ihres Computers können Sie durch „Bearbeiten“ vieler Speicherplätze in Ihrem Rechner vornehmen, ein großer Teil davon liegt im Bereich unter dem Kassettenpuffer, also unterhalb von dezimal 828. (Recorderbuffer $033c...$03fb). Im nächsten Abschnitt geht es weiter mit den verschiedenen Variablen und deren Plätzen. Wo das Betriebssystem usw. liegt, erfahren Sie wie versprochen auch noch, dies aber wirklich erst später, falls Sie es nicht schon selbst wissen.

```
10 INPUT"BEGINN GESCHUETZTER BEREICH";X
20 HB=INT(X/256)
30 LB=INT(X-HB*256)
40 PRINT"HIGHBYTE = "HB
50 PRINT"LOWBYTE  = "LB
60 PRINT"BYTES FREE-MELDUNG WAERE: "HB*256-LB-1-2048

READY.
```

Die Daten (eine Übersicht)

Datenarten

Daten können sowohl als Zeichen oder auch Zahlen interpretiert werden.
Der C64 kennt ebenso wie die anderen Commodore Computer verschiedene Datenarten.

a) Bytes
b) ASCII-Code
c) Bildschirmcode
d) Adressen
e) Integer (Ganzzahlvariable)
f) Gleitkommazahlen
g) Strings

Bytes

Jedes Byte besteht aus 8 Bit, die immer nur einen von zwei Zuständen haben können, nämlich „0" oder „1". Daher sind durch ein Byte die Zahlen 0...255 (= 256 Werte) darstellbar. Die Arbeitsregister des Prozessors, dem im C64 eingebauten 6510A, können jeweils nur ein Byte aufnehmen, man spricht deshalb von einem „8-Bit-Prozessor".

ASCII-Code

Dies ist ein genormter Code, der einem Datensatz von 128 Zeichen ihre bitmäßige Darstellung zuordnet. ASCII = American Standard Code (for) Information Interchange. Das achte Bit wird dabei als Paritätsmerkmal verwendet.
Die Commodore-Rechner weichen allerdings von diesem Standard ab! Dies vor allem deshalb, um beispielsweise auch grafische Zeichen darstellen zu können. Der C64-Zeichensatz umfaßt also 256 Zeichen. Wobei auch noch zwischen Grafik-Modus und Groß-/Kleinschreibung unterschieden werden muß.
Werden beim ASCII die Kleinbuchstaben a...z durch Werte von 97...122 repräsentiert, so geschieht dies beim C64 durch die Zahlen 65...90. Beim Umschalten auf den alternativen Zeichensatz erhält man mit diesen Werten die Großbuchstaben. Diese Abweichungen sind dann wichtig zu wissen, wenn man „fremde" Geräte (z.B. Drucker) ansteuern will.
Man spricht deshalb auch vom CBM-ASCII-Code. Während der Standard-ASCII insgesamt aus 7 Bit besteht, werden beim CBM-ASCII acht Bits verwendet.
Um die Verwirrung dann noch vollends auf den Höhepunkt zu treiben, gibt es dann auch noch den

Bildschirmcode

Dieser entspricht nun leider nicht dem CBM-ASCII! Werden beim C64 ROM-Routinen zur Zeichenausgabe auf den Bildschirm verwendet, so werden diese Zeichen im ASCII-Code an die entsprechenden Routinen übergeben, will man den Bildschirmspeicher aber direkt beschreiben, so müssen diese Zeichen erst in den Bildschirmcode umgewandelt werden. Hierüber gibt aber auch das Handbuch Auskunft, so daß wir hier nicht näher darauf eingehen.
Anmerkung für Umwandlungen zwischen Bildschirm-, ASCII-, CBM-ASCII-Code:
Die Überführung von dem einen in den anderen Code kann durch „Bitmanipulation" der ersten drei Bits erfolgen, denn nur dadurch unterscheiden sie sich.

Adressen

Diese belegen immer 2 Bytes. Da der C64 ja insgesamt 65536 Speicherzellen ansprechen kann, ist die „Adressierung" nur durch minimal 2 Bytes möglich. Hierbei unterscheidet man dann den nieder- und höherwertigen Teil.
Auch andere Bezeichnungen haben sich hierfür eingebürgert:

Low byte = LSB = Last Significant Byte = niederwertiger Teil
High Byte = MSB = Most Significant Byte = höherwertiger Teil

oder anders ausgedrückt, der höherwertige Teil stellt die Seite (Page) dar und der niederwertige Teil die Speicherzellen-Nummer dieser Seite.
Um also die dezimale Adresse einer Speicherstelle zu berechnen, muß man folgendermaßen vorgehen:

LSB + MSB·256

Integer-Zahlen

Hierbei handelt es sich um ganze Zahlen im Bereich von −32768...+32767 mit dem dazugehörigen Vorzeichen. Diese werden prinzipiell durch 2 Bytes dargestellt. Siehe hierzu auch den Abschnitt: Die Variablen.
Gegenüberstellung von Zahlendarstellungen:

Integer	Hex	Dez
00 01	$0001	+1
ff ff	$ffff	−1
00 ff	$00ff	+255
80 00	$8000	−32768

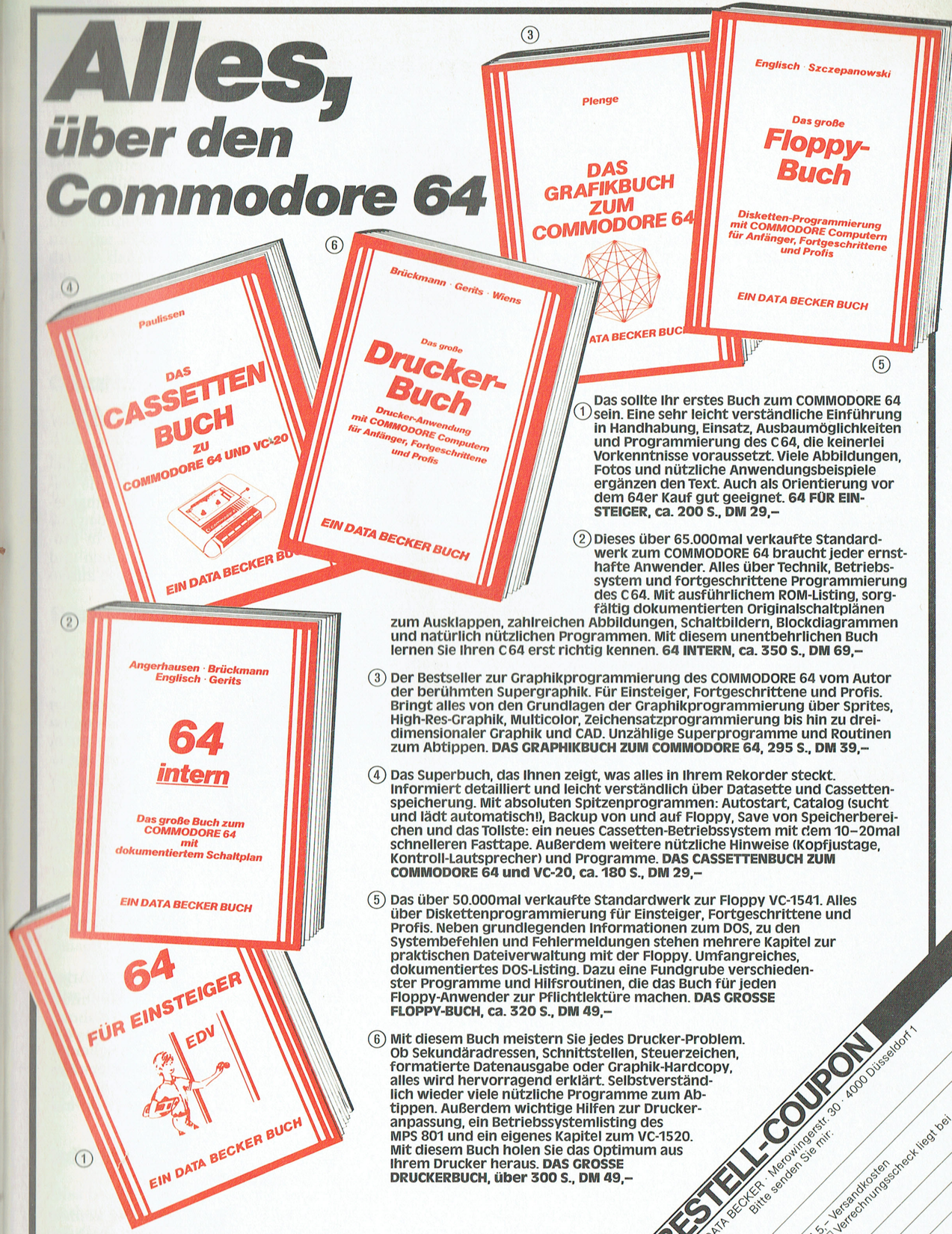
Ganzzahlen (Real-Zahlen) = Gleitkommazahlen

Im Commodore-Basic gibt es zwei mögliche Darstellungsformen: Speicher und Registerformat.
Die Darstellung im Speicherformat geschieht immer durch insgesamt fünf Bytes. Siehe auch hierzu „Die Variablen".
Im Registerformat geschieht die Repräsentation von Gleitkommazahlen immer durch sechs Bytes. Hierbei sind die Bytes eins, drei und fünf denen des Speicherformates gleich. Beim Byte zwei ist im Gegensatz zum Speicherformat, das Vorzeichenbit immer gesetzt, wobei das Byte sechs immer dem Vorzeichenbit des Bytes 2 (Speicherformat) entspricht.

Strings

Als Strings bezeichnet man ASCII-Zeichenfolgen. Sie stellen also alle möglichen Zeichenketten dar. Ein String kann sowohl durch ein einzelnes Zeichen, mehrere, oder aber sogar durch „Nichts" repräsentiert werden. Im letzteren Falle handelt es sich um einen sogenannten „leeren" String. Beispiel: A$="""!
Zu berücksichtigen hierbei ist natürlich auch wieder, daß beim C64 der CBM-ASCII gültig ist.

Alles, über den Commodore 64



① Das sollte Ihr erstes Buch zum COMMODORE 64 sein. Eine sehr leicht verständliche Einführung in Handhabung, Einsatz, Ausbaumöglichkeiten und Programmierung des C64, die keinerlei Vorkenntnisse voraussetzt. Viele Abbildungen, Fotos und nützliche Anwendungsbeispiele ergänzen den Text. Auch als Orientierung vor dem 64er Kauf gut geeignet. **64 FÜR EINSTEIGER, ca. 200 S., DM 29,–**

② Dieses über 65.000mal verkaufte Standardwerk zum COMMODORE 64 braucht jeder ernsthafte Anwender. Alles über Technik, Betriebssystem und fortgeschrittene Programmierung des C64. Mit ausführlichem ROM-Listing, sorgfältig dokumentierten Originalschaltplänen zum Ausklappen, zahlreichen Abbildungen, Schaltbildern, Blockdiagrammen und natürlich nützlichen Programmen. Mit diesem unentbehrlichen Buch lernen Sie Ihren C64 erst richtig kennen. **64 INTERN, ca. 350 S., DM 69,–**

③ Der Bestseller zur Graphikprogrammierung des COMMODORE 64 vom Autor der berühmten Supergraphik. Für Einsteiger, Fortgeschrittene und Profis. Bringt alles von den Grundlagen der Graphikprogrammierung über Sprites, High-Res-Graphik, Multicolor, Zeichensatzprogrammierung bis hin zu dreidimensionaler Graphik und CAD. Unzählige Superprogramme und Routinen zum Abtippen. **DAS GRAPHIKBUCH ZUM COMMODORE 64, 295 S., DM 39,–**

④ Das Superbuch, das Ihnen zeigt, was alles in Ihrem Rekorder steckt. Informiert detailliert und leicht verständlich über Datasette und Cassettenspeicherung. Mit absoluten Spitzenprogrammen: Autostart, Catalog (sucht und lädt automatisch!), Backup von und auf Floppy, Save von Speicherbereichen und das Tollste: ein neues Cassetten-Betriebssystem mit dem 10–20mal schnelleren Fasttape. Außerdem weitere nützliche Hinweise (Kopfjustage, Kontroll-Lautsprecher) und Programme. **DAS CASSETTENBUCH ZUM COMMODORE 64 und VC-20, ca. 180 S., DM 29,–**

⑤ Das über 50.000mal verkaufte Standardwerk zur Floppy VC-1541. Alles über Diskettenprogrammierung für Einsteiger, Fortgeschrittene und Profis. Neben grundlegenden Informationen zum DOS, zu den Systembefehlen und Fehlermeldungen stehen mehrere Kapitel zur praktischen Dateiverwaltung mit der Floppy. Umfangreiches, dokumentiertes DOS-Listing. Dazu eine Fundgrube verschiedenster Programme und Hilfsroutinen, die das Buch für jeden Floppy-Anwender zur Pflichtlektüre machen. **DAS GROSSE FLOPPY-BUCH, ca. 320 S., DM 49,–**

⑥ Mit diesem Buch meistern Sie jedes Drucker-Problem. Ob Sekundäradressen, Schnittstellen, Steuerzeichen, formatierte Datenausgabe oder Graphik-Hardcopy, alles wird hervorragend erklärt. Selbstverständlich wieder viele nützliche Programme zum Abtippen. Außerdem wichtige Hilfen zur Druckeranpassung, ein Betriebssystemlisting des MPS 801 und ein eigenes Kapitel zum VC-1520. Mit diesem Buch holen Sie das Optimum aus Ihrem Drucker heraus. **DAS GROSSE DRUCKERBUCH, über 300 S., DM 49,–**

DATA BECKER
Merowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 310010

BESTELL-COUPON
Einsenden an: DATA BECKER · Merowingerstr. 30 · 4000 Düsseldorf 1
Bitte senden Sie mir:

Name und Adresse bitte deutlich schreiben

zzgl. DM 5,– Versandkosten
☐ per Nachnahme ☐ Verrechnungsscheck liegt bei

Die Variablen

Sie sind eigentlich Symbole, als Stellvertreter der wirklichen Inhalte, wie dies auch bei der Algebra der Fall ist. Manche von Ihnen werden gerade durch die Erwähnung dieses Teilbereiches der Mathematik mit „Schrecken" daran denken, daß es nun um „höheres Rechnen" geht. Aber weit gefehlt. Auch wir überlassen das lieber einem elektronischen Taschenrechner und noch lieber einem Computer.

Wir werden Sie ganz „sanft" und „behutsam" in die „Algebra des Computers", der des C64 einführen, denn die Dinge sind dann, wenn man sie weiß, wirklich sehr, sehr einfach. Sie erfahren auch, wo der C64 diese Stellvertreter hinschreibt, und wie er sie (nicht Sie) verwaltet. Zunächst aber müssen Sie verstehen, was Variablen wirklich sind und welche Arten es gibt.

Fangen wir also nochmal mit der Algebra an. Falls nun schon wieder ein paar Leser zusammenzuckten, ersetzen wir das Wort „Algebra" einfach durch „Rechnen". Bei mathematischen Aufgaben kommt es darauf an, durch Rechenvorgänge, unbekannte Größen zu ermitteln.

Dazu muß man eben die entsprechenden Formeln wissen oder selbst entwickeln. Ein simples Beispiel dafür ist die Rechenaufgabe, wieviel Miete (Gehalt, Lohn, usw.) Sie in einem gewissen Zeitraum von Monaten bekommen oder bezahlen müssen.

Rechnen mit Buchstaben

Um das Problem anfangs nicht zu erschweren, nehmen wir einfach einmal an, daß lediglich 12 Monate sowie Festzahlen vorliegen. Um bei der Miete zu bleiben, die Aufgabe lautet also:

Jahressumme = Betrag monatlich × Anzahl der Monate

oder als Formel ausgedrückt JS = BM·AM, wobei die Abkürzungen nur willkürlich gewählt wurden und aus den Wörtern der allgemein gehaltenen Definition stammen. Die zweite Formel ist also nichts anderes als eine „Buchstabenrechnung", wie die Algebra im Volksmund genannt wird. Aber sie ist doch leicht verständlich, wenn man weiß, um was es geht.
Was wir durch diese Formel erreicht haben, ist nichts anderes, als daß wir die eigentlichen Werte durch „Variablen" dargestellt haben. Sie sehen, die Variablen sind wirklich nur Stellvertreter der echten Werte.

Wir können nun direkt in den Computer eingeben:

print1000·12

Das Ergebnis lautet 12000. Aber dazu hat bestimmt niemand einen Computer (außer dem körpereigenen) eingesetzt. Wir haben nun mit festen Zahlen gerechnet. Wenn wir aber variable Summen haben und auch berechnen wollen, dann kann die Aufgabe auch „frei programmiert" werden.

Taufen von Bekannten und Unbekannten

Vorher aber müssen die Namen der Variablen definiert und dem Computer mitgeteilt werden.

Dies geschieht durch:

AM= 12:BM=1000:JS=BM·AM-:printJS

Wir haben also den Variablen bestimmte Werte zugewiesen, die mathematische Verknüpfung eingegeben und das Ergebnis ausgeben lassen. Das Ganze hätte auch in Programmform geschehen können und hätte dann beispielsweise so ausgesehen:

10 AM = 12
20 BM = 1000
30 JS = BM·AM:printJS

Sie sehen also, vor den Variablen brauchen Sie sich nicht schrecken zu lassen, denn Sie sind es doch, der bestimmt, was sie „wert" sind.

Die Variablentypen

Der C64 unterscheidet verschiedene Variablentypen.

Fließkommavariablen
Ganzzahl- oder Integervariablen
Stringvariablen

und als Variation jeder dieser Arten gibt es noch die indizierten Variablen.
Da der C64 zwischen diesen verschiedenen Gruppen unterscheidet, ist es auch wichtig, die Unterschiede und auch die Kennzeichen, die diese für ihn unterscheidbar machen, zu kennen.

Fließkommavariablen

Diese Art der Stellvertreter können jeden beliebigen Wert innerhalb der Rechengrenzen des Computers

beinhalten oder annehmen. Dies betrifft Zahlen wie z.B.: 1, 1.033, −33.45, 500.6, 9E8 usw.

Der Computer erkennt, daß es sich um eine Fließkommazahl handelt, daran, daß dem Variablennamen kein %- oder $-Zeichen nachgestellt ist. Beispiele für Variablennamen:

X, GG, A1, ZZ ... usw.

Ganzzahlvariablen

Wie der Name schon sagt, haben diese keinen Nachkommateil und sie können lediglich Werte zwischen −32768 und +32767 annehmen. Am Ende ihrer Namensgebung steht immer das Prozentzeichen als Erkennungsmerkmal.

Beispiele: X%, Z2%, H0% ... usw.

Stringvariablen

Als „Wert" haben diese einen beliebigen Text (Zeichen), der meist zwischen Anführungszeichen steht oder über die chr$-Definition zugewiesen wird. Das Kennzeichen dieser Art ist das Zeichen $ (nicht zu verwechseln mit dieser für Sedezimalzahlen ebenfalls verwendeten Kennzeichnung – wie $ffff u.ä.). Dieses Kennzeichen befindet sich am Ende des „Taufnamens".

Beispiele: A$, Z1$, LM$ ... usw.

Die Textvariablen können null bis maximal 255 Zeichen beinhalten. Beispiele hierfür:

N$="" – dies ist eine besondere Definition, der String ist leer, es handelt sich um einen sogenannten „Leerstring".

C$="Computerschau",

CO$="C64", CR$=chr$(13) ... u. ä.

Indizierte Variablen

Dies ist eine Variation, die bei allen vorher genannten Variablenformen möglich ist. Man spricht auch von dimensionierten Variablen, Matrizen, Feldern oder Arrays. Aber keine Angst, es ist weit weniger schwierig, dies zu verstehen, als es im Augenblick aussieht.

Ein einfaches Beispiel aus dem täglichen Leben, macht gleich verständlich, was es damit auf sich hat. Stellen Sie sich bitte einmal ein etwas größeres Haus vor. In diesem gibt es mehrere Stockwerke und in jedem Stockwerk gibt es mehrere Zimmer. Nehmen wir nun einfach einmal an, daß sich auf jeder Etage 4 Zimmer befinden und das Gebäude 5 Stockwerke hat. Um nun eine fremde Person in ein bestimmtes Zimmer zu schicken, könnte man dieser mitteilen, z.B. das Zimmer Nummer 12 aufzusuchen. Voraussetzung dessen aber wäre, daß alle Zimmer durchnumeriert sind oder daß dem „Boten" ein bestimmtes Schema bekannt wäre, durch welches er den richtigen Raum ausfindig machen könnte. Möglich wäre aber auch, die Zimmer jeder Etage mit Nummern von eins bis vier zu kennzeichnen. Diese Art wird in der Praxis tatsächlich so durchgeführt. Denken Sie nur beispielsweise an Gebäude größerer Firmen. Die jeweiligen Zimmernummern beginnen im Parterre mit 0, im ersten Stock mit 1, im 2. Stock mit 2 usw.). Der ausgesandte Bote könnte also auch den Auftrag bekommen, in den dritten Stock zu gehen und dort das Zimmer Nr. 3 aufzusuchen. Damit wäre für ihn eindeutig zu verstehen, welchen Raum er zu betreten hat.

Nun wieder zurück zu unserem Computer, er ist in dieser Hinsicht etwas konsequenter. Er beginnt mit seiner Zählweise immer bei Null. Um auf unser Beispiel zurückzukommen, würde dieser den Zimmern die Ziffern 0...3 zuordnen.

Weshalb er konsequenter ist? Stellen Sie sich doch bitte vor, in einem 20stöckigen Gebäude gäbe es für uns den Raum „00" nur einmal!

Scherz beiseite, wir können nun dem Computer eindeutig sagen, welchen Wert er in einen bestimmten „Raum" legen oder aus einem bestimmten „Raum" holen soll.

Das Ganze in „Computersprache":

W=Z(E,RN)

Im Klartext: Der Wert ist im Zimmer einer gewissen Etage, mit einer gewissen Raum-Nummer. Befinden sich auf dem schon erwähnten Gelände der Firma mehrere Gebäude, und diese Firma befindet sich in einem Industriegebiet mit mehreren Firmen zusammen usw., dann kann dem Computer dies auch mitgeteilt werden.

W=Z(L,S,I,G,E,RN)

Für die Theorie haben wir nun etwas übertrieben, aber die obige Definition würde beispielsweise bedeuten:

Der Wert ist in einem Zimmer, eines bestimmten Landes, einer bestimmten Stadt, eines gewissen Industriegebietes, in einem bestimmten Gebäude, dort in einer bestimmten Etage und in einem Raum mit einer bestimmten Nummer.
Und das alles versteht man unter Array bzw. indizierten Variablen. Nebenbei bemerkt: der C64 hat im Einschaltzustand die Dimensionierung aller Variablen auf elf, nämlich 0...10 begrenzt. Falls dies geändert werden soll, so muß ihm das durch den DIM-Befehl mitgeteilt werden, z.B.:

DIM A(4,13,110)

Wenn Sie diesen Befehl in Ihren Computer eingegeben haben, fragen Sie ihn doch bitte einmal nach dem freien Speicherplatz! Er wird mit 48 antworten. Obwohl wir noch keine Werte zugewiesen haben, hat er uns für unsere Basicprogramme und auch für das direkte Arbeiten nur noch 48 Bytes gelassen. Daß dies selbst für einfache Aufgaben nicht mehr reicht, können Sie durch folgendes Programm feststellen.

10 DIM A(4,13,110)
20 B$="COMPUTERSCHAU"

Ein „run" wird der C64 mit „out of memory" quittieren. Warum dies so ist und auch noch weiteres, werden Sie nun erfahren.

Die Variablen

(Fortsetzung)

Nun wird wieder ein Maschinensprache-Monitor benötigt, denn wir wollen uns ja ansehen, wo und wie die verschiedenen Variablen untergebracht und verwaltet werden. Um alles nun anschaulicher zu machen, haben wir ein Programm eingesetzt, mit dem auch der „Klartext" zu sehen ist.

Zunächst einmal geben wir ein:

```
For i=40928to40959:pokei,0:clr
```

Danach weisen wir den Stringvariablen A$ und B$ die beiden „Worte" zu,

```
A$='COMPUTERSCHAU':
B$='FRANZIS-VERLAG'
```

um anschließend mit dem Monitor nach diesen beiden Worten zu suchen. Wie Sie bereits durch die Pokeadressen erkennen konnten, müssen wir im oberen RAM-Bereich suchen. Das Ergebnis (s. u.) zeigt:

```
M 9FE0 9FFF
:9FE0 00 00 00 00 00 46 52 41 ■■■■■FRA
:9FE8 4E 5A 49 53 2D 56 45 52 NZIS-VER
:9FF0 4C 41 47 43 4F 4D 50 55 LAGCOMPU
:9FF8 54 45 52 53 43 48 41 55 TERSCHAU
```

Die erste Zuweisung, die wir gemacht haben, wurde in den obersten Bereich abgelegt. Die zweite Zuweisung darunter. Wie Sie sehen, sind keinerlei zusätzliche Code oder Zeichen dazwischen. Doch machen wir erst wieder weiter. Definieren Sie bitte A$ neu. Geben Sie ein:

```
A$="★★"
```

und sehen Sie mit dem Monitor wieder nach.

```
M 9FE0 9FFF
:9FE0 00 00 00 2A 2A 46 52 41 ■■■**FRA
:9FE8 4E 5A 49 53 2D 56 45 52 NZIS-VER
:9FF0 4C 41 47 43 4F 4D 50 55 LAGCOMPU
:9FF8 54 45 52 53 43 48 41 55 TERSCHAU

SPEICHERDUMP NACH DER NEUDEFINIERUNG
VON A$
```

Was Sie nun vielleicht etwas verwundert, ist die Tatsache, daß der Inhalt des neuen A$ nun unterhalb dem von B$ liegt und der Inhalt des „alten" Stringes sich immer noch an der alten Stelle im C64 befindet.
Auch die Eingabe von „clr" ändert daran nichts, solange ... ja, solange keine neue Stringzuweisung erfolgt. Hat der C64 vergessen, diesen „Müll" zu beseitigen? Wie kommt er überhaupt klar? Zwischen den einzelnen Textvariablen sind doch keine Trennzeichen!
Falls Sie noch kein „clr" eingegeben haben, gibt er Ihnen die richtigen Antworten auf die Fragen nach den Stringinhalten. Bevor wir Ihnen diese Antworten geben, versuchen Sie es doch einmal selbst.
Deshalb: Geben Sie bitte ein:

```
a=5:b=6
```

Versuchen Sie nun einmal herauszufinden, wo der Computer diese Werte versteckt hat.
Haben Sie es herausgefunden? Wenn ja, dann sind Sie ja schon ein richtiger Profi! Denn ganz so einfach ist es nicht, dies herauszufinden.
Falls nein, dann werden wir es nun wieder ganz langsam erarbeiten. Sie sollten aber vorher unbedingt den

Abschnitt über die Datenarten gelesen haben, da das Wissen hierüber wesentlich zum Verständnis des folgenden sowie auch anderer Teile beiträgt.
Wir beginnen mit den Integer-(Ganzzahl-)Variablen.
Wie Sie dem nachfolgenden Speicherauszug entnehmen können, sieht es im ersten Augenblick so aus, als hätte der Computer mehr getan, als wir eigentlich wollten. Aber er braucht es so, damit er richtig arbeiten kann.

```
A%=5:B%=50:C%=500

M 0800 0820
:0800 00 00 00 C1 80 ■■■■ 00 ■■■■■■■■
:0808 00 00 C2 80 ■■■■ 00 00 ■■■■■■■■
:0810 00 C3 80 ■■■■ 00 00 00 ■■■■■■■■
:0818 00 00 00 00 00 00 00 00 ■■■■■■■■
:0820 00 00 00 00 00 00 00 00 ■■■■■■■■

$0032 = 50
$01F4 = 500

INTERNE ABLAGE VON GANZZAHLVARIABLEN
```

Sie keine überraschende Information mehr darstellen. Es sind dies die Dezimaladressen 45 und 46. Da wir kein Programm im Computer stehen haben, zeigen diese also auf $0803 (=2051). Dort beginnt nun die Information über unsere Variablen. Die Namen hierfür benötigen 2 Bytes. Für A% enthalten deshalb die Adressen $0803 und $0804 die Folge C180. Die Erklärung hierfür ist: Der Buchstabe „A" wird durch chr$(65) dargestellt. Damit das Betriebssystem aber weiß, daß es sich

blen dar und zwar in der Folge High-/Lowbyte, $0005 ist 5. Da es sich um Integervariable handelt, enthalten die drei Folgebytes jeweils die Ziffer Null. Eines ist dabei noch wichtig zu wissen: Das Vorzeichen wird durch das höchstwertige Bit des höherwertigen Bytes repräsentiert!

Alles klar? Versuchen Sie nun mal herauszufinden, was es mit den anderen Variablen, die wir festgelegt hatten, auf sich hat. Geben Sie auch

Doch nun erst der Reihe nach. Pro Integervariable werden 7 Bytes „verbraucht". Der C64 legt diese Variablen – im Gegensatz zu den Stringvariablen – im unteren Basic-Bereich ab und dies oberhalb eines eventuell im Speicher stehenden Basicprogrammes. Daß er für den „Start of Basic Variables" auch wieder Pointer (Zeiger) hat, dürfte für

um eine Integervariable handelt, ist die Zahl 128 hinzuaddiert worden. Die Addition von 65 plus 128 ergibt 193 und dies ist hexadezimal „C1". Da wir die Variable nur mit einem Zeichen (also A anstelle von AA oder AS etc.) festgelegt haben, enthält das zweite Byte $80 und dies ist dezimal 128. Die nächsten beiden Bytes stellen den Wert der Variaeinmal negative Ganzzahlvariable ein und sehen Sie sich die Unterschiede an.

Gleitkommavariable

Dieses Mal ist es ohne Berechnung möglich, die Variable zu finden, denn $41 = chr$(65) = ASCII-Wert

```
A=5:B=50:C=500

M 0800 0820
:0800 00 00 00 41 00 83 20 00 ■■■A■■ ■
:0808 00 00 42 00 86 48 00 00 ■■B■■H■■
:0810 00 43 00 89 7A 00 00 00 ■C■■.■■■
:0818 43 00 89 7A 00 00 00 00 C■■.■■■■
:0820 00 00 00 00 00 00 00 00 ■■■■■■■■

INTERNE ABLAGE VON GLEITKOMMAZAHLEN
```

für „A“, aber was ist aus dem Wert fünf geworden?

Um dies zu verstehen, müssen wir erst einmal zusätzliche Informationen haben.

1. Jede Gleitkommazahl wird durch fünf Bytes plus zwei für den Namen repräsentiert.

2. Das erste Byte (der Zahl!) enthält den Exponenten mit einem Offset von $80, das heißt: Exponent $81 bedeutet Rechtsshift um 1, Exponent $7f bedeutet Linksshift um 1.

3. Die Bytes 2...5 enthalten die Mantisse, wobei das höchstwertige Bit (7) das Vorzeichen darstellt.
So, für die „Mathematiker“ unter Ihnen ist das Ganze nun kein Problem mehr. Den anderen Lesern empfehlen wir, wenn sie tiefer einsteigen wollen, sich noch weiterführendere Literatur zu besorgen, denn einen Kurs in „Mathe“ wollten wir nicht schreiben.

```
A$='COMPUTERSCHAU'

M 0800 0820
:0800 00 00 00 41 80 0D F3 9F ...A.=3.
:0808 00 00 00 00 00 00 00 00 ........

$0041 65
$9FF3 40947

INFORMATION UEBER DIE STRINGVARIABLE A$
```

Stringvariable

Wo sie liegen und wie sie abgelegt werden, haben wir schon vorher gezeigt. Wichtig aber ist noch, woher der C64 weiß, wie er sie finden kann. Die Informationen dazu liegen auch wieder in insgesamt 7 Bytes. Nun können wir uns sehr kurz fassen. Im 1. Byte (also Byte 0) der Name (Teil 1), im 2. Byte der zweite Teil des Namens plus $80. Im nächsten Byte die Länge des Strings und in den beiden Folgebytes die Adresse, wo der String zu finden ist (Low-/Highbyte). Die beiden letzten Bytes enthalten jeweils eine Null.

Zur besseren Verständlichkeit nun eine kleine Übersicht:

Byte:	0	1	2	3	4	5	6	Typ:
	Name 1	Name 2 +$80	String-Länge	Low-Byte	High-Byte	00	00	String-Variable
	Name 1 +$80	Name 2 +$80	High-Byte	Low-Byte	00	00	00	Integer-Variable
	Name 1	Name 2	Exponent +$80		Mantisse (32 Bit)			Gleitkomma-Variable

Nun wissen Sie also, wo der C64 seine Informationen bezüglich der nichtindizierten Variablen, bzw. auch deren Inhalte, herholt.

Es fehlt aber noch die Information, wie es mit den indizierten „Veränderlichen“ aussieht.

Die indizierten Variablen

1

Integer Arrays

Die interne Darstellung haben wir schon kennengelernt, sie entspricht wieder der High-/Lowbyte-Konfiguration. Aber die Abspeicherung selbst erfolgt anders als bei den „normalen“ Integervariablen.

Jedem Array nämlich wird ein sogenannter „Beschreibungsblock“ (Descriptor-) vorangestellt. Dieser enthält die allgemeinen Angaben über das gesamte Array, wobei die Länge dieses Descriptorblocks von der Anzahl und von den Dimensionen abhängt. Um uns dies aber gleich wieder direkt anzusehen, geben Sie bitte ein:
Dim B%(2,3,2)
Wir beginnen gleich wieder mit der Analyse. Ab Speicherstelle $0803 ist die Folge des Descriptorblockes zu sehen.
Dabei ist es wie bei den normalen Integervariablen so, daß die Kennzeichnung in den ersten beiden Blöcken den Namen darstellt. Also C2 80 heißt B%. Byte drei ist das Lowbyte für die Länge des gesamten Feldes (inklusive des Beschreibungsblockes). Byte vier das Highbyte, gefolgt von der Information über die Anzahl der Dimensionen also: drei.
Die Bytes sechs und sieben geben Auskunft über die Anzahl der Arrayelemente für die Dimension „n“, wieder in High-/Lowbytedarstellung.
Die folgenden Bytes stellen die Informationen über die Anzahl der Elemente, jeweils immer für die Dimension „n–x“ (also n–1, n–2, n–3 usw.) dar, bis „n–x“ eins ist.
Direkt dahinter beginnt dann die Abspeicherung der Elemente, jeweils mit zwei Bytes.
Damit alles übersichtlicher wird, sehen Sie sich bitte das Bild oben an.
So, und damit dürfen Sie sich dann selbst weiter durcharbeiten, denn wie man's macht, haben wir damit ja gezeigt.

```
B%(1,2,1)=255:B%(2,1,2)=10

M 0800 0848
:0800 00 00 00 C2 80 53 00 03
:0808 00 03 00 04 00 03 00 00
:0810 00 00 00 00 00 00 00 00
:0818 00 00 00 00 00 00 00 00
:0820 00 00 00 00 00 00 00 00
:0828 00 00 00 00 00 00 00 00
:0830 00 00 00 00 00 FF 00 00
:0838 00 00 00 00 00 00 00 00
:0840 00 00 00 00 00 00 00 00
:0848 00 0A 00 00 00 00 00 00
```

Ganzzahlfelder

Mit dem Wissen all der Dinge, die wir Ihnen schon weiter vorne mitgeteilt haben, sind nun unsere Kurzangaben leicht zu verstehen.
Byte 1 = Name (Teil 1)
Byte 2 = Name (Teil 2)
Byte 3 = Lowbyte (Gesamtlänge des Feldes)
Byte 4 = Highbyte (Gesamtlänge des Feldes)
Byte 5 = Anzahl der Dimensionen
Byte 6 = Highbyte (Anzahl der Elemente, Feld n)
Byte 7 = Lowbyte (Anzahl der Elemente, Feld n)
Byte 8 = Highbyte (Anzahl der Elemente, Feld n–1)
Byte 9 = Lowbyte (Anzahl der Elemente, Feld n–1)
usw. bis „n–2“ = 1!
Direkt dahinter wieder die Elemente selbst, je fünf Bytes lang.

Stringfelder

Hier trifft das gleiche wie bei den Ganzzahlfeldern zu. Ein kleines Beispiel zum Verständnis (s. u.):
Die Stringvariablen selbst finden Sie wieder im oberen Basic-RAM-Bereich.
Den Rest und das ganze „Drumherum“ müssen Sie sich leider selbst erarbeiten, denn wir wollen ja nur die Voraussetzungen schaffen, damit Sie mehr Verständnis für die Arbeitsweise Ihres C64 bekommen.

```
DIMAX$(2,3):AX$(1,1)='COMPUTERSCHAU'
AX$(0,1)='FRANZIS'

M 0800 0838
:0800 00 00 00 41 D8 2D 00 02 ...A.-..
:0808 00 04 00 03 00 00 00 00 ........
:0810 00 00 00 00 00 07 EC 9F .......3
:0818 0D F3 9F 00 00 00 00 00 .3......
:0820 00 00 00 00 00 00 00 00 ........
:0828 00 00 00 00 00 00 00 00 ........
:0830 00 00 00 00 00 FF 00 00 ........

M 9FEC 9FFF
:9FEC 46 52 41 4E 5A 49 53 43 FRANZISC
:9FF4 4F 4D 50 55 54 45 52 53 OMPUTERS
:9FFC 43 48 41 55 94 E3 7B E3 CHAU...
```

Abspeicherung eines Programmes auf Kassette

Da ein Kassettenrecorder nur „Töne" aufzeichnen und auch nur diese wieder einlesen kann, geschieht die Aufzeichnung eines Programmes (auch von Daten) eben durch diese. Dabei liegt die genaue Definition dieser Töne fest.
Darum brauchen Sie sich aber nicht zu kümmern, denn das Betriebssystem erzeugt diese selbst. Was aber für Sie von Interesse sein könnte, ist das Aufzeichnungsschema, denn aus diesem können Sie wiederum Informationen bekommen.

Die Aufzeichnung auf Band geschieht folgendermaßen: Zuerst wird ein Ton zur Synchronisation auf das Band geschrieben. Danach erfolgt die zweimalige (!) Aufzeichnung der Daten. Dies alles geschieht über den Band-(Recorder)puffer, der im Bereich 828...1019 ($033c...03fb) liegt.
Da es sich aber um verschiedene Daten (Programme, Variablenfiles) handeln kann, muß der C64 mitgeteilt bekommen, um was es sich handelt. Außerdem muß er, wenn er ein Programm eines bestimmten Namens einlesen soll, auch erkennen, wann dieses Programm beginnt usw.
Aus diesem Grunde wird jedem File ein „Vorspann" (Header) vorangestellt.

Der Fileheader (Header)

Dieser enthält Informationen über den Filetyp, über evtl. Start und Endadressen usw.

Doch nun wieder der Reihe nach.

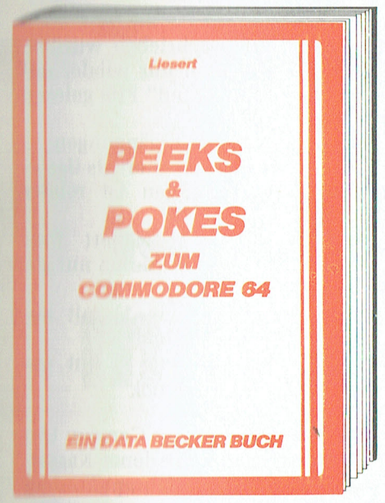
Der C64 liest den Vorspann in den Buffer ein und vergleicht zunächst einmal den Filenamen auf Übereinstimmung. Hat er dann das richtige File erreicht, so beginnt das eigentliche Einlesen des Programmes oder der Daten.
Wie unterscheidet nun der C64 zwischen Daten- und Programmfile usw.? Um dies näher zu untersuchen, schreiben wir zuerst einmal ein kleines Programm und speichern es unter dem Namen „Demo 1" auf Kassette ab.

```
10 REM DEMO FUER PROGRAMMHEADER
20 PRINT"COMPUTERSCHAU ZEIGT WIE MAN'S MACHT!
```

Anschließend bitte gleich nochmals abspeichern mit folgendem Programmnamen:
„Demonstration 2 für die Ablage Programmheader".
Der Programmname sei zu lang und es geht nicht, glauben Sie? Doch, führen Sie es ruhig so durch!
Die Untersuchungen führen wir wieder mittels des (nun hoffentlich durch Sie schon gut beherrschten) Maschinensprachemonitores durch. Laden und starten Sie deshalb bitte Ihren „Monitor" und kehren Sie in den Basicmodus zurück. Prinzipiell würde ein Programm auch im Monitorbetrieb eingelesen werden können, da aber die meisten Monitore immer „absolut" laden, müssen wir wieder zurück ins Basic. (Daß Sie Ihren Monitor nach der Installation wieder mit „sys 2047" starten können, setzen wir einfach voraus.)
Nun starten Sie den Lesevorgang vom Kassettenrecorder. Wenn der Bildschirm zeigt: found PROGRAMMNAME, drücken Sie bitte einfach die „RUN/STOP"-Taste.
Nun springen Sie bitte wieder in den Monitor und lassen Sie sich den Bereich $033c...$0354 ausgeben.
Sie erkennen nun bereits, in diesem Bereich befinden sich „Daten". Mittels unseres eingesetzten Monitors ergibt sich folgendes Bild:

```
M 033C 0350
:033C 01 01 08 4D 08 44 45 4D ...M.DEM
:0344 4F 20 31 20 20 20 20 20 20 0 1
:034C 20 20 20 20 20 20 20 20
```


Maschinensprache



Kein Problem!

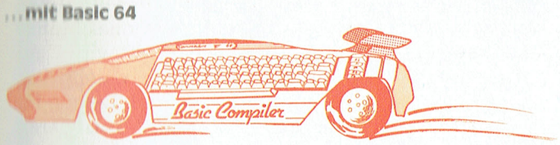
PEEKS & POKES ZUM COMMODORE 64
Mit den wichtigen PEEK und POKE Befehlen lassen sich auch von BASIC aus viele Dinge tun, für die sonst komplizierte Maschinenroutinen nötig wären. Dieses Buch erklärt leichtverständlich den Umgang mit PEEKs und POKEs. Mit einer riesigen Anzahl wichtiger POKEs und ihren Anwendungsmöglichkeiten. Dazu wird der Aufbau des C-64 prima erklärt: Betriebssystem, Interpreter, Zeropage, Pointer und Stacks, Charakter-Generator, Sprite-Register, Programmierung von Schnittstellen, Abschalten des Interrupts. Dazu eine erste Einführung in die Maschinensprache. Viele Beispielprogramme.
PEEKS & POKES FÜR DEN COMMODORE 64, ca. 170 Seiten, DM 29,–

MASCHINENSPRACHEBUCH
Der über 40.000mal verkaufte Bestseller, der Sie leicht verständlich und umfassend in die Maschinensprache einführt. Sie lernen Aufbau und Arbeitsweise des 6510 Mikroprozessors kennen, erfahren wie man Maschinenprogramme eingibt und startet und lernen den Umgang mit Monitor, Assembler und Disassembler. Assembler und Disassembler sind im Buch als Programm ebenso enthalten, wie ein Einzelschrittsimulator, mit dem Sie Ihre Programme schrittweise ausführen können. Viele ausführlich beschriebene Beispielprogramme und Routinen. Mit diesem bewährten Buch sollte Ihnen der Einstieg in die Maschinensprache nicht schwerfallen. **DAS MASCHINENSPRACHEBUCH ZUM COMMODORE 64, ca. 200 Seiten, DM 39,–**

MASCHINENSPRACHE FÜR FORTGESCHRITTENE
Sie haben den Einstieg in die Maschinensprache geschafft? Dann zeigt Ihnen der „neue Englisch", wie Sie jetzt ein Profi werden. Von der Problemanalyse bis zum Maschinensprachealgorithmus werden Sie umfassend in die Grundlagen der professionellen Maschinenspracheprogrammierung eingeführt... Dazu wieder viele Beispielprogramme, komplette Maschinenroutinen und wichtige Tips & Tricks zur Maschinenprogrammierung und zur Arbeit mit dem Betriebssystem. **DAS MASCHINENSPRACHEBUCH FÜR FORTGESCHRITTENE, ca. 200 Seiten, DM 39,–**

Machen Sie Ihrem C 64 Beine...

...mit Basic 64



Der Compiler BASIC 64 bietet die Möglichkeit, BASIC-Programme entweder in Maschinensprache oder in einen sogenannten Speedcode zu übersetzen. Beide Varianten sorgen dafür, daß ihre Programme 4 bis 14mal schneller laufen. Bearbeiten Sie mit BASIC 64 alle Programme, die Ihnen schon immer zu langsam waren. Mit dem kompakten Speedcode können Sie den Speicherplatzbedarf Ihres Programms um 25% verringern, während der speicherplatzaufwendigere Maschinencode zusätzlichen Geschwindigkeitszuwachs bringt. BASIC 64 kann jedes Programm verarbeiten, das im COMMODORE 64 BASIC geschrieben wurde (Ausnahme: einzelne POKE-Befehle) und unterstützt teilweise auch die bekannten Befehlserweiterungen. Außerdem können Sie mit BASIC 64 den Speicherplatz für Daten um 24 K erweitern. Nebenbei erledigt BASIC 64 einige Arbeiten für Sie: Umformung mathematischer Ausdrücke, möglichst ökonomische Speicherplatzausnutzung und Integer Arithmetik. Durch eine völlig veränderte Stringbehandlung schrumpft die gefürchtete Garbage Collection auf wenige Sekunden. Alle Optionen werden per Menue aufgerufen und Eingaben auf ihre Korrektheit überprüft. Mit BASIC 64 haben Sie ein Hilfsmittel in der Hand, das Ihre BASIC-Programme schneller macht, als Sie es bisher für möglich gehalten haben. **DM 99,–**

...mit Profimat

Wer sich tiefer in die Innereien des Computers begeben will, kommt ohne besonderes Werkzeug nicht aus. Einerseits muß der volle Einblick in alle Speicherbereiche möglich sein, andererseits soll der Umgang mit Maschinenprogrammen so komfortabel wie möglich gestaltet sein.
PROFIMAT hat Lösungen für beide Probleme: Der Maschinensprache-Monitor PROFI-MON bietet alle Hilfsmittel zum Umgang mit Maschinenprogrammen; PROFI-ASS ist ein Macro-Assembler, der das Schreiben von Maschinenprogrammen fast so einfach macht wie das Programmieren in BASIC.

PROFIMAT in Stichworten:
Registerinhalte und Flags anzeigen – Speicherinhalte anzeigen – Maschinenprogramme laden, ausführen und speichern – Speicherbereiche durchsuchen, vergleichen, füllen und verschieben – echter Einzelschrittmodus – Setzen von Unterbrechungspunkten – schneller Trace-Modus – Rückkehr zu BASIC – formatfreie Eingabe – Verkettung beliebig vieler Quellprogramme – erzeugter Objektcode kann in Speicher oder auf Diskette gehen – formatiertes Assemblerlisting – ladbare Symboltabellen – redefinierbare Symbole – Operatoren – Unterstützung der Fließkommaarithmetik – bedingte Assemblierung – Assemblerschleifen – MACROS mit beliebigen Parametern.
DM 99,–

DATA BECKER
Merowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 310010

BESTELL-COUPON
Einsenden an: DATA BECKER · Merowingerstr. 30 · 4000 Düsseldorf 1
Bitte senden Sie mir:
zzgl. DM 5,– Versandkosten
☐ per Nachnahme ☐ Verrechnungsscheck liegt bei
Name und Adresse bitte deutlich schreiben

Dabei steht im ersten Byte das Kennzeichen für den Filetyp, nämlich eine Eins und dies heißt für den Computer, daß es sich um ein Basicprogramm handelt, welches relativ geladen wird; d.h. es wird immer dorthin geladen, wo die Zeiger für Basicbeginn hinzeigen.
Das zweite und das dritte Byte geben Auskunft über die Programmstart-, das vierte und fünfte Byte über die Programmende-Adresse.
In unserem Falle also:

Programmbeginn: $0801
Programmende: $084d
In den darauffolgenden Bytes ist der Programmname („Demo 1“) zu finden.

Programme länger als 16 Zeichen

Nun laden Sie bitte das zweite Programm, welches Sie auf die Kassette „gesaved“ haben und sehen Sie sich bitte wieder den Bereich von $033c...$036c an. Der überlange Programmname ist da!
Sie haben also die Möglichkeit, auch längere Programmnamen abzuspeichern und, und das ist nun das Interessante daran, den Inhalt dieser „Überlängen“ z.B. gleich zur „Copyrightabfrage“ auszunützen.
Ja, selbst die Ablage von kleinen Maschinenroutine-Daten ist denkbar, allerdings ist das nicht ganz so einfach!
Doch nun wieder zurück zum eigentlichen Thema.
Bisher hatten wir in der Speicherstelle $033c eine „Eins“ stehen, weil es sich um ein Programm handelte, welches mit dem normalen Basic-Modus abgespeichert wurde. Nun speichern wir das gleiche Programm aus dem Monitor heraus ab. Dies geht je nach Monitor unterschiedlich, deswegen hier 2 Möglichkeiten, wie es klappen könnte:
S ”test” 01 0801 084d
oder
S ”test”,01,0801,084d
also entweder mit oder ohne Komma. Das „absolute“ Abspeichern kann auch von Basic aus, mit:
save„name“,1,1
erreicht werden.
Hat es dann geklappt, dann das Programm wieder einladen. Das Nachsehen in der Speicherzelle $033c zeigt diesmal eine „Drei“ als Zeichen dafür, daß es ein „absolutes“ Programm ist.
Die Wirkungen der beiden Programmbeispiele erkennen Sie am besten, wenn Sie den Basicstart verschieben. Sie wissen doch hoffentlich noch, wie das geht?
„poke 44,x:new“ (Die Null nicht vergessen!).
So, und dann laden Sie bitte doch mal das Programm „Demo 1“; wie Sie sehen, läßt es sich sofort listen, es wurde also an den neuen Basicstart verschoben und dies, obwohl die Beginnadresse ($033d/033e) auf $0801 zeigte.
Nun das gleiche mit dem Programm, welches mittels des Monitors auf Kassette geschrieben wurde. Der Befehl „list“ zeigt einen leeren Bildschirm.
Poken Sie nun bitte wieder um: „poke 44.8“ und lassen Sie nun „listen“. Das Programm ist zu sehen. Nun dürfte der Unterschied des Inhaltes von $033a ja wohl klar sein.
Das mit dem Monitor „abgespeicherte“ wurde durch die Verschiebung des Basicbeginnes nicht „berührt“. Es lud sich dort ein, wo es auch ursprünglich war.
Das erste Byte hat also folgende Bedeutung:
„1“ = Basic-Programm, wird ab Basicstart geladen
„2“ = Datenblock, die folgenden Bytes stellen die Daten dar
„3“ = Maschinenprogramm, wird immer absolut geladen
„4“ = Datenheader, es folgt ein Daten-, (kein Programm-)File.
„5“ = EOT End of Tapeblock; Kennzeichnung von Bandende.
Beim Filetyp 2 handelt es sich also um Daten, welche mit „print“ File auf das Band geschrieben wurden. Diese können auch nur wieder mit „get“ File oder „input“ File gelesen werden.
Nun noch ein paar Ergänzungen:
1) save„name“,1 speichert als Basicprogramm (Einlesen ist relativ möglich)
2) save„name“,1,1 speichert Programme absolut (Einlesen nur absolut)
3) save„name“,1,2 wie 1) mit zusätzlichem EOT-Block
4) save„name“,1,3 wie 2) mit zusätzlichem EOT-Block

Nun zu der Essenz des Ganzen: Um die Adressen verschiedener Kassettenprogramme zu erfahren, können Sie sich dies entweder mit einem Monitor ansehen, oder aber mit folgendem kleinen Programm erfahren:

```
100 OPEN1,1,0,""
110 FORI=1TO4
120 A(I)=PEEK(828+I)
130 NEXTI
140 PRINT
150 PRINT"BEGINN: "A(1)+256*A(2)
160 PRINT"ENDE  : "A(3)+256*A(4)
170 CLOSE1
```

So, zum Schluß nun nochmals auf die anfangs gemachte Mitteilung, daß ein Programm zweimal hintereinander abgespeichert wird, zurückkommend. Dies geschieht deshalb, damit Ihr C64 evtl. Lese- oder kleine Bandfehler noch korrigieren kann. Rein theoretisch kann man also das Laden eines Programmes bereits nach der Hälfte abbrechen und dann trotz der Fehlermeldung, die durch den Abbruch erzeugt wurde, auflisten bzw. sogar starten. Am besten hören Sie sich ein Programm einmal mit einem normalen Kassettenrecorder an, dann wissen Sie auch, wie sich die ganze Aufzeichnung anhört. Sie lernen dann auch sehr gut den Header und auch Daten zu unterscheiden. Es ist sogar möglich, Datenfiles mittels eines „fremden“ Headers zu öffnen und trotzdem die Daten korrekt einzulesen. Vielleicht denken Sie mal darüber nach, wie dies geschehen könnte.

Erzeugung von „neuen“ Basiczeilen mittels Tastaturpuffer

Der C64 hat seinen Tastaturpuffer im Bereich $0277...$0280. In diesem Bereich werden die Tastatureingaben abgelegt.
Man kann also bis zu 10 Zeichen „vorschreiben“. Daß dies so ist, läßt sich leicht mit einem kleinen Beispiel aufzeigen. Sie brauchen nur eine etwas längere „for... next...“-Schleife einzugeben und anschließend die Zahlenfolge 0...9 „blind“ einzutippen.
Nach Beendigung der Schleife sehen Sie dann die eingegebenen Zeichen auf dem Schirm und rechts davon den Cursor. Diese Schleife können Sie sowohl direkt eingeben, als auch als Programm laufen lassen.
(Der C64 verhält sich dabei etwas anders als sein „Bruder“ CBM 3032; dieser (3032) fängt bei Überschreitung von 10 Zeichen einfach wieder bei 0 an!)

```
100 INPUT"MAXIMALE EINGABELAENGE";X$
110 IFVAL(X$)<0ORVAL(X$)>9THEN100
120 PRINT"LAENGE DES PUFFERS     :"VAL(X$)
130 FORI=1TO5000:NEXTI
140 PRINT"ANZAHL DER ZEICHEN WAR:"PEEK(198)
150 POKE198,VAL(X$)

READY.

PROGRAMM ZUR DEMO TASTATURPUFFER
```

Blindtastung

Während des Programmlaufes kann dann aber die Anzahl der „Blindtastung“ ohne weiteres wieder geändert werden.
Dies geschieht durch poke198,x: wobei x ein Wert zwischen 0 und 9 sein soll. Tippen Sie ruhig das kleine Programm in Ihren C64 ein und spielen Sie etwas damit.
Es klappt ganz gut, wie Sie sehen können. Durch diese Methode kann man Eingaben, die über „get“ file eingelesen werden sollen, völlig streichen bis zur wirklichen Eingabe.
Das heißt, Spielereien auf der Tastatur während eines Programmlaufes können dann keinen Unsinn ergeben.

```
100 OPEN9,0
110 PRINT"GEBEN SIE NUN SOVIELE ZEICHEN EIN,  WIE SIE WOLLEN"
120 FORI=1TO2000:NEXTI
130 GET#9,A$,B$,C$,D$
140 PRINTA$,B$,C$,D$
150 PRINT"AB NUN NICHTS MEHR EINGEBEN
160 FORI=1TO1000:NEXTI
170 POKE198,0
180 PRINT"NUN BITTE DIE EINGABEN
190 FORI=1TO4
200 GET#9,A$(I)
210 IFA$(I)=""THEN200
220 NEXTI
230 FORI=1TO4
240 PRINTA$(I):NEXT
250 CLOSE9

PROGRAMM TASTATURPUFFER 2
```

Lassen Sie dieses Programm bitte ein paarmal „laufen". Was der Pokebefehl bewirkt, sehen Sie am besten, wenn Sie ihn (Zeile 170) einmal entfernen (kann durch Voranstellen von REM geschehen) und dann das Programm wieder laufen lassen.

Automatische Erzeugung von Basiczeilen

Dieses Wissen über den Tastaturpuffer und seine Arbeitsweise ermöglichen uns nun, dies in einer völlig anderen Art auszunützen. Stellen Sie sich bitte vor, Sie möchten eine kleine Datei aufbauen. Die Daten sollen hierbei in Datastatements abgelegt werden. Normalerweise schreibt man also die entsprechenden Zeilen-Nummern und gibt die Daten ein. Eine völlig andere Methode zeigen wir nun auf.
Hier geschieht die Aufnahme der Daten völlig automatisch während des Programmlaufes.
Ganz korrekt aber ist diese Aussage nicht, denn in Wirklichkeit wird das Programm immer wieder beendet, aber unsere in den Tastaturpuffer geschriebenen Zeichen bewirken, daß das Programm immer wieder neu startet:
Hier das Programm:

```
100 PRINT"■":PRINT:PRINT:PRINT:PRINT:PRINT
110 A=1000
120 INPUT"EINGABE";A$:PRINT"■
130 IFA$="ENDE"THENEND
140 PRINT"■■■■"A"DATA"A$
150 PRINT"A="A;"+10"
170 PRINT:PRINT
180 PRINT"G┌120
190 PRINT"■"
200 FORI=631TO633:POKEI,13:NEXT
210 POKE198,3

AUTOMATIK-DATA-PROGRAMM
```

Mit diesem Programm können nun automatisch die Eingaben als Datas abgelegt werden. Die Eingabe von „ENDE" bewirkt, daß der Automatiklauf wieder unterbrochen wird. Dieses Programm läßt sich natürlich auch erweitern.
So ist es durchaus denkbar, durch diese Methode z.B. einen „Datagenerator", oder auch ein Reassemblierprogramm zu realisieren. Gegebenenfalls kann das Programm ja auch an einen anderen Speicherplatz eingelesen werden.
Es muß ja nicht immer der normale Basicbeginn sein. Sie sehen, dieser kleine Trick kann sehr „starke" Auswirkungen haben.
Was wurde eigentlich gemacht, damit dies alles funktionierte? Wenn Sie das Programm näher ansehen, haben wir in Zeile 180 den Befehl „goto 120" ausgeben lassen. Durch Zeile 200 haben wir in den Tastaturpuffer 3mal die Taste „Return" eingeschrieben und in Zeile 210 haben wir dem Computer mitgeteilt, daß sich drei Zeichen im Tastaturpuffer befinden. Den Rest sehen Sie ja selbst im Programm und können sicherlich nun schon alleine die Analyse durchführen.

Automatisches Einlesen von Kassette und automatisches Speichern auf Disk

Derjenige, der anfangs nur die Datasette für seinen C64 hatte und sich dann später eine Diskettenstation zulegte, hat üblicherweise einiges zu tun, wenn er all seine Kassettenprogramme auf „Disk" bringen will. Auch ihm kann ein kleines Programm helfen und den Vorgang automatisieren.

Das ganze geht – aber nur ohne weiteres für normale Basicprogramme – mit dem nachfolgenden Programm. Auch hier wird der im letzten Abschnitt bereits „mißbrauchte" Tastaturpuffer wieder zweckentfremdet. Wichtig beim Start des Programmes ist, daß die „Playtaste" des Recorders vor „run" gedrückt wird. Das Programm könnte man natürlich auch noch erweitern, um Diskettenprogramme automatisch auf Kassette zu „bringen". Die Vorgehensweise müßte dann eine Mischung aus der Methodik der automatischen Datenübernahme (Abloge der Diskettenprogrammnamen in Datas), dann Löschen des bereits auf Band gesaveten Programmnamens aus den Datastatements usw., sein.
Ein derartiges Programm wäre vor allem für denjenigen von Interesse, der einem Bekannten einige Programme auf Kassette geben will. Für Sicherheitskopien auf Kassette gibt es bessere und schnellere Methoden. Doch nun das Programm, das wir „RECDISK-SCHAUFEL" genannt haben:

```
100 PRINT"■┌■■■■■"
110 PRINT"FORX=0TO15:Y=PEEK(1150+X):P┌1390+X,Y:N¯"
120 PRINT"■■5■"CHR$(34)"§0:                    "CHR$(34)",8"
130 PRINT"■■■■┌"CHR$(34)"RECDISK-SCHAUFEL"CHR$(34)",8"
140 PRINT"■■■■RUN"
150 POKE198,7:POKE631,19:FORR=1TO7:POKE631+R,13:NEXT:END



RECDISK-SCHAUFEL
```

Automatische Entfernung von REM-Zeilen

Bei der Erstellung von Programmen ist es äußerst wichtig, diese mit sogenannten „REMarks" zu versehen, denn diese sind es, bzw. sollten es sein, die ein Basicprogramm erläutern.

Hat man dann endlich das Programm fertig erstellt und es ist fehlerfrei, dann verbrauchen gerade diese Erklärungen – wenn viel „bemerkt" wurde – sehr viel Speicherplatz. Bei großen Programmen kann es sogar vorkommen, daß der Speicherplatz so knapp wird, daß der Programmlauf statt der erwarteten Ergebnisse ein „out of memory" bringt.
Abhilfe kann die Entfernung der Zeilen mit den Erklärungen schaffen.

Der Computer kann es schneller

In einem gut dokumentierten Programm kann dies allerdings eine sehr zeitaufwendige Angelegenheit sein. Deswegen kann derjenige, der ausreichendes Wissen über den Tastaturpuffer hat, sich sehr schnell und komfortabel mittels eines kleinen Programmes helfen.
Voraussetzung für das Gelingen des Entfernungsvorganges ist allerdings, daß die REMs unmittelbar nach der Zeilennummer stehen. Befinden sie sich nicht als erstes „Statement" in einer Zeile, werden diese durch das nachfolgende Programm auch nicht entfernt. Diesen Effekt kann man insofern ausnützen, daß nur bestimmte Zeilen entfernt werden. Bei der Programmerstellung braucht man also nur darauf zu achten, daß im Programm zu verbleibende REMs erst nach einem weiteren Zeichen (z. B. einem Doppelpunkt) stehen.
Das Hilfsprogramm selbst kann, wenn entsprechende „Utilities" zur Verfügung stehen, über „merge" oder „append" an das im Speicher stehende zu „Bearbeitende" angehängt werden. Wenn nicht, muß dieses eben ganz normal eingetippt und mit Run 63500 gestartet werden. Anschließend dann das Hilfsprogramm löschen und das „entremark'te" Hauptprogramm wieder abspeichern.

Erklärungen zum Programm „REM-Entferner":

In Zeile 63500 wird zuerst der Bildschirm gelöscht und dann der Adreßmerker (AM) auf Basicbeginn gesetzt.
In Zeile 63510 wird in der Variablen LZ (Linkzeiger) die nächste Zeilennummer abgelegt. Außerdem wird die aktuelle Zeilennummer (ZN) berechnet und die zu prüfende Speicheradresse in PA (Prüfadresse) abgespeichert.
In Zeile 63520 wird überprüft, ob die Startzeile des REM-Entferners erreicht ist und wenn ja, der Programmlauf beendet. Zeile 63530 verzweigt nach Zeile 63510, wenn in der Prüfadresse kein REM steht. Außerdem wird der Adreßmerker auf die Linkzeile „gesetzt".
Die Zeile 63540 bewirkt, daß die „REM"-beinhaltende Zeilennummer in die ersten beiden Bytes des Recorderpuffers gepoked wird. Anschließend wird der Bildschirm gelöscht und die Zeilennummer auf dem Schirm ausgegeben. Poke 198,10 teilt dem C64 mit, daß der Tastaturpuffer 10 Zeichen enthält.
Zeile 63550 enthält die in den Tastaturpuffer zu pokenden Datas, liest diese, schreibt sie in den Tastaturpuffer und beendet das Programm.
Da aber die Datas nichts anderes sind, als der Befehl „home" und Return mit dem anschließenden Befehl „goto 63560" und einem abschließenden return, startet das Programm wieder mit der Zeile 63560 und dort wird der Adreßmerker geholt und wieder nach 63510 verzweigt.
Sie können sich ja nun weitere Hilfsroutinen ausdenken, die ebenfalls mittels Tastaturpuffer und Befehlsübernahme vom Bildschirm arbeiten.

```
63490 REM REM-ENTFERNER
63500 PRINT"■":AM=PEEK(44)*256+PEEK(43)
63510 LZ=PEEK(AM+1)*256+PEEK(AM):ZN=PEEK(AM+3)*256+PEEK(AM+2):PA=AM+4
63520 IFZN=63490THENEND
63530 IFPEEK(PA)<>143THENAM=LZ:GOTO63510
63540 POKE828,INT(AM/256):POKE829,AM-(PEEK(828)*256):PRINT"■"ZN:POKE198,10
63550 DATA19,13,71,207,54,51,53,54,48,13:FORI=631TO640:READD:POKEI,D:NEXT:END
63560 AM=PEEK(828)*256+PEEK(829):GOTO63510

READY.
```


Änderung der „Characterget"-Routine

Immer wieder können wir, um mehr Verständnis für unseren, „nun doch schon etwas sezierten" Computer zu erlangen, einen Maschinensprachemonitor einsetzen. So auch diesmal wieder. Es geht darum, eigene kleine Routinen, die der C64 nicht hat, in seinen „Wortschatz" einzubinden.

Dazu müssen wir uns aber erst darüber klar werden, wie er die einzelnen Basic-Statements holt und ausführt.

Da wir nicht voraussetzen, daß jeder Leser in Maschinensprache fit ist, beschränken wir uns deswegen auf das wichtigste.

Woher/Wohin

Im Betriebssystem-ROM des C64 befindet sich unter anderen auch die Characterget-Routine.
Diese Routine, die in den technischen Unterlagen auch den „symbolischen" Namen CHRGET oder CHARGET trägt, wird – bei einem Einschaltreset – aus dem ROM-Bereich ($e3a2) in den RAM-Bereich ($73...$8a) herunterkopiert, da zwei Bytes dieser Routine laufend verändert werden müssen. Dies wäre in einem ROM-(Read Only Memory-) Speicher ja nicht möglich.

Was macht die „Characterset"

Sinn dieser Routine ist, die Verbindung zwischen dem Interpreter und Basic herzustellen.
Wenn unter Basic gearbeitet wird, dann prüft diese Routine auf Doppelpunkt und Leerzeichen (Space). Aber es steckt noch mehr in dieser kleinen aber sehr effektvollen Routine. Sie erkennt nämlich auch, ob im Direkt- oder Basicmodus gearbeitet wird.
Um dieses nun besser zu verstehen, sollten Sie sich bitte das Bild unten näher ansehen.

```
,0073 E6 7A     INC $7A
,0075 D0 02     BNE $0079
,0077 E6 7B     INC $7B
,0079 AD .. ..  TEXTZEIGER
,007C C9 3A     CMP #$3A
,007E B0 0A     BCS $008A
,0080 C9 20     CMP #$20
,0082 F0 EF     BEQ $0073
,0084 38        SEC
,0085 E9 30     SBC #$30
,0087 38        SEC
,0088 E9 D0     SBC #$D0
,008A 60        RTS

CHARGET-ROUTINE
```

Nun zur Erklärung:
Zuerst wird durch den Befehl INC $7A das niederwertige Byte des Adreßzeigers erhöht, der dann dadurch auf das nächste zu holende Zeichen gerichtet ist. Dieses Zeichen wird dann in den Akkumulator geladen und daraufhin verglichen, ob es sich um einen Doppelpunkt (hexadezimal 3A) handelt (CMP #$3A). Ist der Akkuinhalt größer oder gleich dem ASCII-Wert des Doppelpunktes, dann wird zu dem in $8a stehenden Befehl RTS (Return Subroutine) verzweigt, der einen Rücksprung aus der Unterroutine – genauso wie in Basic – bewirkt.
Übrigens endet der Ziffernbereich des ASCII-Codes mit $39! Falls aber der Akkuinhalt kleiner als $3a (also maximal $39) sein sollte, dann wird die Verzweigung zum Rücksprungbefehl nicht durchgeführt, sondern durch CMP #$20 überprüft, ob es sich um ein Leerzeichen (Space = chr$(32) = $20) handelt. Falls dies zutrifft, dann erfolgt wieder eine Verzweigung auf den Beginn der Charget-Routine, der Textzeiger wird erhöht usw.
Falls die letzte Überprüfung aber ergibt, daß es kein Leerzeichen ist, erfolgen nun zwei Subtraktionen mit den Werten $30 und $d0.
Doch hier wird es nun interessant. War nämlich der Akkuinhalt kleiner als $30 (also kleiner als der Code für die Ziffer 0), dann wird das „Carry-Bit" gelöscht und gleich wieder gesetzt. War es aber eine Zahl ($30...$39 = 0...9), dann wird die „Carry-Flag" durch die zweite Subtraktion gelöscht. Der in Maschi-

nensprachepro grammierung fortgeschrittene Leser wird nun schon erkannt haben, daß durch diese beiden Subtraktionen eigentlich überhaupt nichts subtrahiert wurde, denn #$30 plus #$d0 ergibt #$100, da aber der höchste darstellbare Wert mit 8 Bit #$ff ist, wurde also #$00 abgezogen. Es wurde nur die Carry-Flagge entsprechend beeinflußt. Genau dadurch aber kann der C64-Interpreter zwischen Direkt- und Basicmodus unterscheiden. Ganz schön trickreich, was?
Aber nun wieder zurück zu unserem eigentlichen Ausgangspunkt. Die Überschrift dieses Abschnittes heißt ja: Veränderung der Charget-Routine und soll uns ermöglichen, eigene kleine oder auch größere Basicerweiterungen in den Befehlssatz einzubauen.

```
D 0073 008A
,0073 E6 7A     INC $7A
,0075 D0 02     BNE $0079
,0077 E6 7B     INC $7B
,0079 AD 02 02  LDA $0202
,007C 4C 3C 03  JMP $033C
,007F 00        BRK
,0080 C9 20     CMP #$20
,0082 F0 EF     BEQ $0073
,0084 38        SEC
,0085 E9 30     SBC #$30
,0087 38        SEC
,0088 E9 D0     SBC #$D0
,008A 60        RTS

DISASSEMBLIERTE CHRGET-ROUTINE
(GEAENDERT)
```

Nun ändern wir

Direkt nach dem Befehl lade Akku mit Textzeiger treiben wir unseren eigenen kleinen Programmkeil ein. Wir springen nämlich direkt nach diesem Befehl auf unsere eigene Routine. Dies geschieht in unserem Beispiel durch den Befehl JMP $033c. Das heißt, wir springen in eine vorher eigens geschriebene Maschinenroutine, die im Kassettenrecorderbuffer liegt. Sie könnte natürlich auch in einem anderen Speicherbereich (so z. B. auch von $c000...cfff) liegen, wichtig ist lediglich, daß dieser Bereich nicht durch ein z. B. laufendes Programm überschrieben wird. Die disassemblierte Chrget-Routine zeigt, daß nach dem eingetriebenen Keil nun der Befehl BRK steht und die ursprüngliche Routine völlig normal weitergeht.
Eine Bitte gleich vorweg. Versuchen Sie bitte nicht gleich mittels des Monitors diese Änderungen durchzuführen, es könnte für Sie eine unangenehme Überraschung werden. Denn bei entsprechendem Inhalt des Kassettenpuffers würde der C64, nach dem Ausstieg aus dem Monitor evtl. total „verrückt" spielen. Wenn Sie Glück haben, trifft der Adreßzeiger beim Abarbeiten der Charget und der Verzweigung, ohne viel Veränderungen im Speicher durchzuführen, auf einen BRK-Befehl und „landet" dann wieder im Monitor, da dieser den sogenannten Break-Vektor auf sich selbst gerichtet hat. Wir wollen uns aber doch nicht auf den Zufall verlassen.
Dieser Sprung nach $033c geschieht natürlich erst durch Betätigung der Return-Taste, denn erst dann tritt die Routine bei $0073 in Aktion. Schreiben Sie bitte, wenn Sie sich diesen Effekt etwas näher ansehen wollen, in die Speicherstelle $033c eine $00. So nun können Sie aus Ihrem Monitorbetrieb aussteigen.

```
M 033C 034F
:033C C9 3E D0 08 A9 93 20 D2 FF 4C 74 A4 C9 3A B0 03
:0344 FF 4C 74 A4 C9 3A B0 03 4C 80 00 60 00 00 00 00

ZUSATZBEFEHL
```

Monitorbetrieb – forever?

Betätigen Sie also ruhig die Taste x und drücken dann RETURN, es sieht so aus, als wäre alles normal. Aber alles, was Sie nun eingeben, führt nur dazu, daß Sie wieder im Monitor landen. Nochmals zur Erinnerung: Der Monitor hat beim ersten Aufruf die Zeiger für den BRK-Vektor auf sich selbst gerichtet und dadurch, daß in $033c die Hex-Zahl $00 steht, wird er immer wieder aufgerufen, sobald die Charget-Routine dorthin umgelenkt wird.
Sinn und Zweck einer Veränderung aber sollte ja nicht sein, daß Sie immer wieder in den Monitorbetrieb zurückkehren, sondern wir wollen ja eine eigene Basicerweiterung durchführen. Deshalb müssen wir vorher in den Kassettenpuffer auch etwas sinnvolles hineinschreiben.
Aus diesem Grunde geben Sie bitte das im Bild Zusatzbefehl angegebene kleine Programm ein. Mit dem Monitor ist dies sicherlich schnell geschehen.

Das disassemblierte Listing hierfür sieht folgendermaßen aus:

Grundlagen, Teil 12

```
D 033C
,033C C9 3E     CMP #$3E
,033E D0 08     BNE $0348
,0340 A9 93     LDA #$93
,0342 20 D2 FF  JSR $FFD2
,0345 4C 74 A4  JMP $A474
,0348 C9 3A     CMP #$3A
,034A B0 03     BCS $034F
,034C 4C 80 00  JMP $0080
,034F 60        RTS

DISASSEMBLIERTES LISTING:
ZUSATZBEFEHL
```

Was macht nun dieses Programm? Gehen wir wieder Schritt für Schritt vor.
In $033c steht der Befehl cmp #$3E. Das heißt, daß abgefragt wird, ob es sich um eine „spitze Klammer rechts", also das Zeichen, das Sie erhalten, wenn Sie Shift und Punkt gleichzeitig drücken, handelt. Ist dies nicht der Fall, dann wird nach $0348 verzweigt, dort steht ein Teil der alten Charget-Routine und dann wird wieder in die Charget-Routine, nämlich nach $80 gesprungen und ganz normal, so als wäre nichts gewesen, weitergemacht. Wurde aber das nämliche Zeichen eingegeben, dann wird der Akku mit #$93 geladen und die Ausgaberoutine (Bsout) angesprungen, das heißt der Akkuinhalt wird „geschrieben". Da #$93 aber der Code für CLS, also für „mache den Bildschirm frei" ist, wird immer dann, wenn die nun erweiterte CHRGET-Routine darauf trifft, der Bildschirm gelöscht. Um dies nun zu testen, müssen Sie wieder aus dem Monitor-Betrieb aussteigen und dann geben Sie doch bitte folgendes ein:

10rem > 30 RETURN

Wie Sie sehen konnten, hat sich nichts getan, die Zeile 10 wurde übernommen, wie sonst auch. Haben wir also einen Fehler gemacht? Geben Sie bitte nun einmal LIST ein, Sie sehen, die Zeile 10 enthält unser gewünschtes Zeichen. Also RUN...AHA..., den Effekt haben Sie nun ja selbst gesehen. Daß es auch im Direktmodus klappt, können Sie durch Eingabe „geshifteter Punkt" und Return selbst überprüfen.

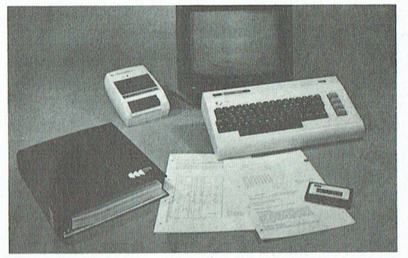
Nun können Sie zusätzliche Basicbefehle einbauen

So, wir hoffen, daß Sie nun nicht nur etwas darüber erfahren haben, wie man eigene Zusatzbefehle einbauen kann, sondern auch darüber, wie die Charget-Routine arbeitet und wann Sie „aktiv" wird.
Überraschen Sie doch mal jemanden damit, daß durch Eingabe eines von Ihnen definierten Zeichens der Computer einen Systemreset durchführt!
Wie dies geschehen kann? Ganz einfach, ändern Sie den Befehl JSR $ffd2 einfach ab in $fce2, fertig.
Aber ärgern Sie Ihre Mitmenschen und sich selbst nicht zu sehr damit. Übrigens ist nach diesem Reset Ihr Maschinensprache-Monitor (wir hoffen, daß Sie einen haben, der nicht bei Basicstart beginnt) noch erhalten und Sie können ihn wie gewohnt aktivieren. Leider aber wurde der Kassettenpuffer gelöscht und auch die ersten Bytes eines Basicprogrammes sind geändert.
Im nächsten Abschnitt erfahren Sie nun, wie man eine Hardcopy-Funktion in den Befehlssatz einbaut und natürlich auch noch weiteres.

Programme
Anwendungen
Aufgaben
Kompakt-Kurs
BASIC

Erst durch fundierte Kenntnisse der Programmiersprache BASIC wird ein Computer zu einem leistungsfähigen Helfer – denn ohne Programm ist ein Computer nutzlos.

Der Christiani Kompakt-Kurs BASIC vermittelt Schritt für Schritt und leicht verständlich, wie man Probleme in Programme umsetzt und wie sie verwirklicht werden.



Der Kurs ist in drei Teile gegliedert und besteht aus dem 200 Seiten umfassenden Kursmaterial und einer lehrgangsbegleitenden Tonbandkassette.

Am Ende des Kurses können Sie den Christiani Test BASIC machen – wir bestätigen Ihnen dann Ihre Kenntnisse in Form eines Zertifikats.

Der Lehrgang kostet DM 198,–.

Christiani
Fortbildung

Technisches Lehrinstitut
Postfach 3519164 · 7750 Konstanz
in Österreich: Ferntechnikum Bregenz

Coupon auf Postkarte aufkleben oder im Umschlag einsenden.
Sie erhalten sofort kostenlos ausführliches Informationsmaterial über den Lehrgang BASIC.

Name, Vorname

Straße, Nr.

PLZ, Ort 19164

Hardcopy, ein simples kleines Programm

Leider hat der C64 keine eingebaute Routine, die es ermöglichen würde, den Bildschirm sofort auszudrucken. Für manche Anwendungen aber wäre dies nicht nur wünschenswert, sondern sogar unbedingt erforderlich.

```
1000 ; HARDCOPY 1 DRUCKT KOMPLETTEN BILDSCHIRM
1010 *=$C000;SOURCEADDRESS
1020 &=$C000;QUELLADDRESS
1030 :          LDA #$7F        ;FILEPARAMETER SETZEN
1040 :          LDX #$04        ;GERAETENUMMER UND
1050 :          LDY #$00        ;SEKUNDAERADRESSE
1060 :          JSR $FFBA       ;PARAMETER SETZEN
1070 :
1080 :          LDA #$00        ;DATEINAME 0
1090 :          JSR $FFBD       ;FILENAMENPARAMETER SETZEN
1100 :
1110 :          JSR $FFC0       ;OPEN
1120 :          BCS ENDE
1130 :          LDA #$00        ;LB BILDSCHIRM
1140 :          STA $FD
1150 :          LDA #$04        ;HB BILDSCHIRM
1160 :          STA $FE
1170 :          LDX #$7F
1180 :          JSR $FFC9       ;CKOUT AUSGABEGERAET SETZEN
1190 :          LDX #$19        ;ZEILENANZAHL
1200 REIHE      LDA #$0D
1210 :          JSR $FFD2       ;BSOUT
1220 :          JSR $FFE1       ;STOP-TASTE ABFRAGEN
1230 :          BEQ ENDE
1240 :          LDY #$00
1250 SPALTE     LDA ($FD),Y
1260 :          STA $FC         ;CODEWANDLUNG
1270 :          AND #$3F
1280 :          ASL $FC
1290 :          BIT $FC
1300 :          BPL MARKE1
1310 :          ORA #$80
1320 MARKE1     BVS MARKE2
1330 :          ORA #$40
1340 MARKE2     JSR $FFD2       ;BSOUT
1350 :          INY
1360 :          CPY #$28
1370 :          BNE SPALTE
1380 :          TYA
1390 :          CLC
1400 :          ADC $FD
1410 :          STA $FD
1420 :          BCC MARKE3
1430 :          INC $FE
1440 MARKE3     DEX
1450 :          BNE REIHE
1460 :          LDA #$0D
1470 :          JSR $FFD2       ;BSOUT
1480 :          JSR $FFCC       ;CLRCH EIN-/AUSGABE RUECKSETZEN
1490 ENDE       LDA #$7F
1500 :          JMP $FFC3       ;FILE SCHLIESSEN
1510 ;END
READY.
ASSEMBLERLISTING FUER HARDCOPY 1
```

Eigentlich ist ein solches Programm gar nicht so groß, deswegen verwundert es uns schon ein wenig, daß etwas derartiges bei den Commodore-Computern nicht „eingebaut" ist. Aber was soll's. Hier nun eine Hardcopy-Routine, die im Speicherbereich ab $c000 liegt und welche gleichzeitig bei Aktivierung den Zusatzbefehl „Pfeil nach links" – dies entspricht dem Aufruf der Funktion – in den Basicbefehlssatz einbindet. Wir haben es uns dieses Mal sehr leicht gemacht und Ihnen das Ganze als kommentiertes Assemblerlisting nachstehend aufgeführt.

Die wenigsten von Ihnen haben wahrscheinlich einen Assembler, der mit diesem „Programm" etwas anfangen kann. Aber wir wollen ja nicht unfair sein, denn Sie wollen ja evtl. ab und zu eine Hardcopy-Funktion benutzen.

Deswegen und auch noch aus anderen Gründen haben wir uns etwas einfallen lassen, was unserer Meinung nach schon lange auf dem Markt für den C64 gefehlt hat. Doch davon im nächsten Abschnitt, wenn wir wieder einmal zeigen, wie man's macht. Das Assemblerlisting ist an den wichtigen Punkten kommentiert, so daß wir uns eine detaillierte Erklärung ersparen. Es ist ja auch nicht jedermanns Sache, in Maschinensprache zu programmieren, aber es ist höchst interessant. Wir müssen das immer wieder feststellen.

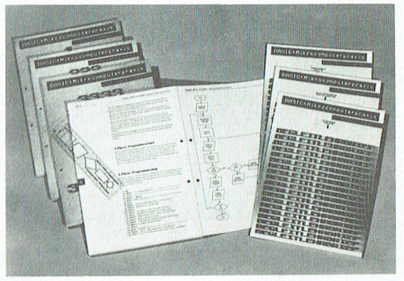

```
1000 ; HARDCOPY 2 KPL BILDSCHIRM, RAND 9 AM ENDE
1010 *=$C000;SOURCEADDRESS
1020 &=$C000;QUELLADDRESS
1025 :          LDA #$09
1026 :          STA ZAEHLER
1030 :          LDA #$7F        ;FILEPARAMETER SETZEN
1040 :          LDX #$04        ;GERAETENUMMER UND
1050 :          LDY #$00        ;SEKUNDAERADRESSE
1060 :          JSR $FFBA       ;PARAMETER SETZEN
1070 :
1080 :          LDA #$00        ;DATEINAME 0
1090 :          JSR $FFBD       ;FILENAMENPARAMETER SETZEN
1100 :
1110 :          JSR $FFC0       ;OPEN
1120 :          BCS ENDE
1130 :          LDA #$00        ;LB BILDSCHIRM
1140 :          STA $FD
1150 :          LDA #$04        ;HB BILDSCHIRM
1160 :          STA $FE
1170 :          LDX #$7F
1180 :          JSR $FFC9       ;CKOUT AUSGABEGERAET SETZEN
1190 :          LDX #$19        ;ZEILENANZAHL
1200 REIHE      LDA #$0D:JSR $FFD2
1210 :          LDA #$1D
1220 :          LDY ZAEHLER
1230 RAND       JSR $FFD2
1240 :          DEY
1250 :          BNE RAND
1260 :          JSR $FFD2       ;BSOUT
1270 :          JSR $FFE1       ;STOP-TASTE ABFRAGEN
1280 :          BEQ ENDE
1290 :          LDY #$00
1300 SPALTE     LDA ($FD),Y
1310 :          STA $FC         ;CODEWANDLUNG
1320 :          AND #$3F
1330 :          ASL $FC
1340 :          BIT $FC
1350 :          BPL MARKE1
1360 :          ORA #$80
1370 MARKE1     BVS MARKE2
1380 :          ORA #$40
1390 MARKE2     JSR $FFD2       ;BSOUT
1400 :          INY
1410 :          CPY #$28
1420 :          BNE SPALTE
1430 :          TYA
1440 :          CLC
1450 :          ADC $FD
1460 :          STA $FD
1470 :          BCC MARKE3
1480 :          INC $FE
1490 MARKE3     DEX
1500 :          BNE REIHE
1510 :          LDA #$0D
1520 :          JSR $FFD2       ;BSOUT
1530 :          JSR $FFCC       ;CLRCH EIN-/AUSGABE RUECKSETZEN
1540 ENDE       LDA #$7F
1550 :          JMP $FFC3       ;FILE SCHLIESSEN
1560 ZAEHLER    NOP
1570 ;END
READY.
ASSEMBLERLISTING HARDCOPY 2
```

So und nun für diejenigen, die die Hardcopy-Routine implementieren möchten, abschließend ein Programm, das diese Routine beinhaltet und zwar als Datalisting.

Wenn dieses Programm fehlerfrei abgelaufen ist, braucht nur mit Sys12*4096 (oder Sys49152) aktiviert werden und schon kann man nach Herzenslust Bildschirmkopien drucken. Allerdings haben wir uns auf eine relativ einfache Ausführung zum Ausdruck beschränkt, es wird weder normale Grafik, noch hochauflösende Grafik ausgedruckt. Diese Routinen für „harte Kopien" sind nämlich auch sehr stark Druckerabhängig, das heißt, nicht jeder anschließbare Drucker versteht die gleiche Sprache. Deswegen also die Beschränkung unsererseits. Übrigens arbeitet die Routine mit dem MPS 802 einwandfrei und dies dürfte auch für den 1526 zutreffen.

Praxis-
orientiertes
Programmieren
BASIC +
Mikrocomputer-
praxis

Eigene Programme in BASIC erstellen, fremde Programme verstehen und umschreiben, gekaufte Standardprogramme beurteilen und einsetzen, Aufbau und Funktionsweise eines Mikrocomputers kennen, die wichtigsten Anwendungen beherrschen und über ein solides theoretisch-praktisches EDV-Grundwissen verfügen. Das alles vermittelt Ihnen der Lehrgang BASIC + Mikrocomputerpraxis.



Hauptfächer: Programmieren in BASIC — Programmiertechniken — Hardware.
Anwendungsfächer: Programmierstudien — Kaufm.-kommerzielle Anwendungen — Technisch-wissenschaftl. Anwendungen — Computer-Graphik — Musik und Spiele.
Nebenfächer: Betriebssysteme — Programmiersprachen — Daten und ihre Darstellung — Evaluation von Mikrocomputern — Mathematische Grundlagen.

Der Lehrgang umfaßt 14 Lieferungen zu je DM 122,–.

Christiani
Fortbildung

Technisches Lehrinstitut
Postfach 3519164 · 7750 Konstanz
in Österreich: Ferntechnikum Bregenz

Coupon auf Postkarte aufkleben oder im Umschlag einsenden.
Sie erhalten sofort kostenlos ausführliches Informationsmaterial über den Lehrgang BASIC + Mikrocomputerpraxis.

Name, Vorname

Straße, Nr.

PLZ, Ort 19164

```
1000 ; HARDCOPY 1 DRUCKT KOMPLETTEN BILDSCHIRM
1010 *=$C000;SOURCEADDRESS
1020 &=$C000;QUELLADDRESS
1030 ; KEIL IN CHRGET
1040 :          SEI
1050 :          LDA #$4C        ;JMP-BEFEHL IN CHRGET
1060 :          STA $7C         ;BRINGEN
1070 :          LDA #$0F        ;DIESER ZEIGT
1080 :          STA $7D         ;AUF ERWEITERTE
1090 :          LDA #$C0        ;CHRGET-ROUTINE
1100 :          STA $7E
1110 :          CLI
1120 :          RTS             ;FERTIG
1130 ;ERWEITERTE CHRGET
1140 :
1150 CHRGET2    CMP #$5F        ;AUF PFEIL NACH LINKS PRUEFEN
1160 :          BNE DPVER       ;KEIN PFEIL NACH LINKS
1170 :          JSR HARDCOPY    ;AUSDRUCKEN
1180 :          JMP $A474       ;READY AUSGEBEN UND IN DIREKTMODUS
1190 DPVER      CMP #$3A        ;IST ES EIN DOPPELPUNKT
1200 :          BCS END1
1210 :          JMP $0080       ;WEITERMACHEN IN ALTER CHRGET
1220 END1       RTS
1230 ;HARDCOPY EINSPRUNG
1240 HARDCOPY   LDA #$7F        ;FILEPARAMETER SETZEN
1250 :          LDA #$7F        ;FILEPARAMETER SETZEN
1260 :          LDX #$04        ;GERAETENUMMER UND
1270 :          LDY #$00        ;SEKUNDAERADRESSE
1280 :          JSR $FFBA       ;PARAMETER SETZEN
1290 :
1300 :          LDA #$00        ;DATEINAME 0
1310 :          JSR $FFBD       ;FILENAMENPARAMETER SETZEN
1320 :
1330 :          JSR $FFC0       ;OPEN
1340 :          BCS ENDE
1350 :          LDA #$00        ;LB BILDSCHIRM
1360 :          STA $FD
1370 :          LDA #$04        ;HB BILDSCHIRM
1380 :          STA $FE
1390 :          LDX #$7F
1400 :          JSR $FFC9       ;CKOUT AUSGABEGERAET SETZEN
1410 :          LDX #$19        ;ZEILENANZAHL
1420 REIHE      LDA #$0D
1430 :          JSR $FFD2       ;BSOUT
1440 :          JSR $FFE1       ;STOP-TASTE ABFRAGEN
1450 :          BEQ ENDE
1460 :          LDY #$00
1470 SPALTE     LDA ($FD),Y
1480 :          STA $FC         ;CODEWANDLUNG
1490 :          AND #$3F
1500 :          ASL $FC
1510 :          BIT $FC
1520 :          BPL MARKE1
1530 :          ORA #$80
1540 MARKE1     BVS MARKE2
1550 :          ORA #$40
1560 MARKE2     JSR $FFD2       ;BSOUT
1570 :          INY
1580 :          CPY #$28
1590 :          BNE SPALTE
1600 :          TYA
1610 :          CLC
1620 :          ADC $FD
1630 :          STA $FD
1640 :          BCC MARKE3
1650 :          INC $FE
1660 MARKE3     DEX
1670 :          BNE REIHE
1680 :          LDA #$0D        ;CARRIAGE RETURN
1690 :          JSR $FFD2       ;BSOUT
1700 :          JSR $FFCC       ;CLRCH EIN-/AUSGABE RUECKSETZEN
1710 ENDE       LDA #$7F
1720 :          JMP $FFC3       ;FILE SCHLIESSEN
1730 ;END
READY.

                                HARDCOPY 1 MIT KEIL IN CHARACTERGET-ROUTINE
```

```
500 REM HARDCOPY MIT ←
501 AA= 49152 :EA= 49285
502 FORI=AATOEA
503 READD$
504 IFLEFT$(D$,1)=CHR$(42)THENP$=MID$(D$,2):P=VAL(P$):ZZ=ZZ+1
505 IFPTHENIFS<>PTHENPRINT"FEHLER IN ZEILE: "1000+ZZ:STOP
506 IFPTHENS=0:I=I+1:IFI=>EATHENPRINT"FERTIG":END
507 IFPTHENP=0:S=0:I=I-2:NEXTI
508 D=VAL(D$):S=S+D
509 PRINTCHR$(147)"ADRESSE  STRING  DATEN  PEEK  ZEILE
510 PRINTITAB(9)D$TAB(16)D;:POKEI,D:PRINTTAB(23)PEEK(I)TAB(29)ZZ+1000:NEXTI
1001 DATA120,169,076,133,124,169,015,133,125,169,192,133,126,088,096,* 1868
1002 DATA201,095,208,006,032,033,192,076,116,164,201,058,176,003,076,* 1637
1003 DATA128,000,096,169,127,169,127,162,004,160,000,032,186,255,169,* 1784
1004 DATA000,032,189,255,032,192,255,176,074,169,000,133,253,169,004,* 1933
1005 DATA133,254,162,127,032,201,255,162,025,169,013,032,210,255,032,* 2062
1006 DATA225,255,240,049,160,000,177,253,133,252,041,063,006,252,036,* 2142
1007 DATA252,016,002,009,128,112,002,009,064,032,210,255,200,192,040,* 1523
1008 DATA208,230,152,024,101,253,133,253,144,002,230,254,202,208,205,* 2599
1009 DATA169,013,032,210,255,032,204,255,169,127,076,195,255,000,* 1992
READY.
```


Der Standard-Data-Generator

Haben Sie auch schon einmal verzweifelt versucht, ein in einer Zeitschrift abgedrucktes Maschinenprogramm, welches als Datastatementausführung gelistet war, zum „Laufen" zu bringen? Ist es Ihnen nicht immer gelungen?

Haben Sie Stunden damit verbracht, den (die?) Tipp- oder Lesefehler zu finden?

Ja? ... Dann befinden Sie sich in überaus guter Gesellschaft. Nicht immer war es die Schuld des „Abtippers", wenn ein Programm trotz genauester Vergleiche mit dem Original nicht das tat, was es eigentlich tun sollte. Ärgerlich ist es natürlich auch, wenn sich der Computer dann beim Start des Programmes durch einen Fehler „aufhängte" usw.
Ein Grund für derartige Effekte liegt beispielsweise daran, daß Datastatements mit einer Schreibmaschine abgeschrieben werden. Oder daß sich beim „Setzen" der Programme im Verlag Übertragungsfehler einschleichen.
Was aber hilft es, wenn man am Programmlaufende freundlicherweise durch den Computer die Mitteilung bekommt, daß ein Fehler aufgetreten sei?
Meist nicht viel, vor allem dann, wenn es sich um seitenlange Listings handelt.
In vielen Fällen ist man dazu übergegangen, in diese Listings nach etlichen Zeilen, sogenannte Blockprüfsummen einzubauen, um dem geplagten Leser zu helfen.
Wir gehen wesentlich weiter, wir haben ein Programm entwickelt, welches sich in einigen kleinen, aber sehr, sehr positiv auswirkenden Punkten, von all diesen anderen, uns bisher bekannten Datagenerator unterscheidet. Deswegen nennen wir dieses Programm:

Standard-Data-Generator

Beim Programm für die Hardcopy-Funktion konnten Sie bereits ein mit diesem Generator erzeugtes Programmlisting kennenlernen. Wahrscheinlich ist Ihnen dabei schon aufgefallen, wie übersichtlich die Datastatements untereinanderstehen und daß am Ende einer jeden Zeile die Prüfsumme steht. Vielleicht aber ist Ihnen beim Abtippen ein kleiner Fehler passiert und beim Programmlauf hat Ihnen Ihr C64 freundlich mitgeteilt, in welcher Zeile sich der Fehler befindet. Die Fehlersuche hat sich dadurch bestimmt auf ein Minimum reduziert. Derjenige, der das Programm abtippt, kann die führenden Nullen ruhig weglassen, er verliert dann zwar in seinem eigenen Programm an Schönheit, aber auf die Funktion selbst hat es keinen Einfluß.
Doch nun zum Programm selbst:
In Zeile 20 wird festgelegt, daß die eigentlichen Datastatements mit Zeile 1001 beginnen. Dies dient beim „Lauf" des verbleibenden „Pokegenerators" zur einfachen Fehlerzeilenerkennung (a=1000). Weiterhin wird in Zeile 10 auch noch die Anzahl der Datas pro Zeile festgelegt, um mit verschiedenen Druckern (mit unterschiedlichen Druckzeilenbreiten) arbeiten zu können. AZ ist dabei die Anzahl der eigentlichen gelesenen Daten – minus 1 – pro Zeile.
Durch Zeile 40 wird mitgeteilt, daß es sich um die Version für den C64 handelt. Durch geringfügige Modifikation der Zeilen 260 und 300 kann dieser Generator auch an die anderen Commodore-Computer angepaßt werden.
Die Zeilen 60...80 dienen zur Abfrage und Eingabe verschiedener Daten, welche dann auch in den „Poke-Generator" übernommen werden.
Die Zeile 90 gibt dann bereits die erste neue Zeile, die neu erzeugt wird, aus. Die Zeilennummer beträgt in diesem Falle 500 und enthält als Zeileninhalt hinter dem „rem" den File-Namen. Die nachfolgende Zeile (100) erzeugt eine neue Zeile mit der Nummer 501 und den Informationen bezüglich der Anfangs- und Endadresse. Außerdem wird noch eine Leerzeile (enthält nur einen Doppelpunkt) zur besseren Übersichtlichkeit generiert (Zeilen-Nummer 1000).
In 120 wird lediglich der Zähler für die Peek-Adressen gesetzt. Die Zeile 130 bewirkt, daß die für den Programmlauf erforderlichen Daten auf den Bildschirm geschrieben werden. Zeile 140 bewirkt die Ausgabe von „GOTO 150" und springt dann zu Zeile 260.
Die Zeile 150 ist dann die eigentliche Startzeile des Generators, welche über die entsprechenden Bildschirmausgaben und -übernahmen immer wieder mit „goto" angesprungen wird. Das Goto wird deshalb benutzt, weil ein Programmstart mit Run alle Variablen wieder löschen würde.
Da der Einsprung in diese Zeile nach jeder erzeugten Datazeile erfolgt, wird dimensioniert, die Prüfsumme (PS) auf Null gesetzt, abgefragt, ob die Endadresse erreicht ist, und wenn nicht, dann der entsprechende „Peekwert" in Y eingelesen. Danach wird der Adressenzähler erhöht (z=z+1). Dann erfolgt die Prüfsummenaddition (ps=ps+y) und zu Zeile 170 gesprungen. Zeile 160 wird nur dann abgearbeitet, wenn die Endadresse erreicht wur-

de und dient nur zur Zwischenspeicherung von Werten zwecks Ausstieg aus der For-next-Schleife (fori =0toaz) in Zeile 150, anschließend wird zu Zeile 190 gesprungen.
Zeile 170 dient zur Erzeugung des Datastrings und unterdrückt das führende Space (die Mid$-Funktion).
In Zeile 180 werden abhängig von der Länge des Strings noch „Nullen“ vorangestellt, falls die Ausgangszahl nur ein- bzw. zweistellig war.
Zeile 190 schließt die For-next-Schleife und löscht den Bildschirm.
Durch Zeile 200 wird die Data-Zeile erzeugt und auf den Schirm ausgegeben. Bei erreichter Endadresse hat die Variable AZ den Wert der Laufvariablen I aus Zeile 160.
Zeile 210 schließt die in 200 begonnene Schleife.
Durch Zeile 230 wird bewirkt, daß die letzte Prüfsumme ausgegeben wird, der Bildschirm mit dem Sprung nach 270 beschrieben, aber dann nach 260 gesprungen wird.
Zeile 240 erzeugt die jeweiligen Prüfsummenausgaben am Ende der Datazeilen, mit Ausnahme der Letzten. Außerdem werden, wie schon für Zeile-Nr. 130 beschrieben, die erforderlichen Variablen auf den Schirm geschrieben.

Zeile 250 schreibt das erforderliche „GOTO 150“ auf den Bildschirm.

Die Zeile 260 ebenso wie 300, sind die eigentlich wichtigsten Zeilen dieses Programmes überhaupt, denn durch diese wird das „dynamische Keyboard“ verwirklicht. Zunächst wird durch printchr$(19) der Cursor in „Homestellung“ gebracht. Dann wird einige Male die Zahl 13 in den Tastaturpuffer eingeschrieben und dies ist nichts anderes als der Code für die Return-Taste. Unmittelbar danach bewirkt das poke 198,x, daß dem C64 mitgeteilt wird, daß sich in seinem Tastaturpuffer x Zeichen befinden. Da sich der Cursor in der linken oberen Ecke des Bildschirmes befindet, werden nun durch diese Simulation, die auf dem Bildschirm stehenden Zeilen, Wertdefinitionen und Goto-Befehle übernommen. Der Aussprung aus dem Programm wurde durch Zeile 230 eingeleitet. Denn dort erfolgte die Bildschirmausgabe „goto270“ und in dieser und den folgenden Zeilen werden nun die für den übrigbleibenden „Poke-Generator“ nicht mehr erforderlichen „Hilfszeilen“ gelöscht.
Aus dem alten Generatorprogramm verbleiben dann die Zeilen 502...510 und die weiteren wurden ja neu erzeugt.

Den Rest nun noch in Kürze:

502 startet die Lese-Pokeschleife
In 504 wird abgefragt, ob es sich bei dem durch Zeile 503 gelesenen String um den Prüfstring (Kennzeichen: *) handelt. Falls ja, dann wird die Prüfsumme gebildet (p) und der Zeilenzähler (zz) um eins erhöht.
Falls eine Prüfsumme (p) vorliegt, muß der letztgelesene String der Prüfstring sein und damit der letzte String einer Zeile. Falls nun die Prüfsumme (p) nicht mit der in Zeile 508 gewonnenen Summe der Daten (s) übereinstimmt, muß ein Fehler vorliegen und Zeile 505 bricht den Programmlauf ab.
Zeile 506 setzt die Summe der Daten für die neue einzulesende Datazeile auf Null, erhöht den Schleifenzähler für die Schleife Anfangsadresse bis Endadresse und frägt ab, ob das Programm zu beenden ist.
507 setzt die Prüfsumme (p) und die Summenvariable auf Null, erniedrigt die Schleife um zwei (in 506 wurde um eins erhöht!) und fährt mit der Schleife fort.
In 508 wird der zu pokende Datenwert aus dem gelesenen String gebildet.
Die Zeilen 509 und 510 zeigen dann den ganzen Vorgang des Ablaufes auf dem Bildschirm an, wobei in 510 auch der Pokebefehl steht. Damit ist das ganze Programm beschrieben, aber ein paar Anmerkungen sollen noch folgen:
Wegen der Einerschritte der Datazeilen ist für den „Abtipper“ eine leichte Überprüfung auf fehlende Zeilen möglich. Die dreistellige Ausgabe der Ursprungs-Programm-Datas erfolgte wegen des besseren „Bildes“ und auch zur besseren Lesbarkeit. Die Prüfsummen wurden zwecks sofortigen Erkennens mit einem „★“ versehen und brauchen nicht unbedingt eingegeben zu werden, allerdings kann dann auch keine ordnungsgemäße Fehlermeldung erfolgen.
Wie Sie sehen konnten, kamen wir beim Listing ohne die – leider nicht immer klar zu identifizierenden – grafischen Steuerzeichen aus.
Auf eine Gesamtprüfsumme haben wir verzichtet, da ja jede Zeile einzeln geprüft wird.
Einen Selbststart des gepokten Programmes haben wir unterlassen, da die Anfangsadresse ja nicht immer mit der Einsprung- bzw. Startadresse übereinstimmen muß und außerdem nicht immer gewünscht wird.
Ein Compilieren des Programmes ist nicht möglich, da beim „Generatorlauf“ ja dauernd neu Basiczeilen erzeugt werden. Aber eine erweiterte modifizierte Version, welche dauernd auch den Basicstart umpoket, könnte durchaus bei entsprechendem Programmaufbau compiliert und dadurch schneller werden.
Es sollte aber ja ein Standard-Programm sein, welches als Ausgangsbasis für noch weitergehende Versionen dienen kann. Außerdem wollten wir von der Computer-Schau nur zeigen, wie man’s macht!
Wir sind noch weiter gegangen. Für viele Interessierte ist der Umgang mit einem Maschinensprachemonitor so geläufig, daß es diesen lieber wäre, die Programme in Hexzahlen vorliegen zu haben, um sie direkt eingeben zu können. Auch für diese Leser haben wir gesorgt und den Standard-Data-Generator noch weiter entwickelt, so daß wir beide Anhängerscharen (Dezimal- und Hex-Fans) „beglücken“ können. Eigentlich würde ja eine Ausführung reichen, aber derjenige, der mit den Hexzahlen nicht umgehen kann, hat natürlich Probleme, wenn er Fehler suchen und erkennen soll. Deswegen also beide Arten!

Nachdem wir die Beschreibung beim Standard-Data-Generator relativ ausführlich gehalten haben, kann sich jeder ja den „Hex-Standard-Data-Generator“ selbst analysieren. Die Arbeitsweise ist genauso, lediglich die entsprechenden Zahlenumwandlungen usw. werden dabei zusätzlich durchgeführt.

Viel Spaß beim Data-Generieren!


```
10 REM DEFINITIONEN UND FESTLEGUNGEN
20 A=1000:AZ=14:REM A=DATAZEILENBEGINN -1 : AZ=ANZAHL DER PRG-DATAS PRO ZEILE
30 :
40 PRINTCHR$(147)"COMPUTERSCHAU STANDARD-DATAGENERATOR":PRINT"FUER C 64
50 :
60 INPUT"FILE-NAME";F$
70 INPUT"ANFANGSADRESSE (DEZIMAL) ";AA
80 INPUT"ENDADRESSE     (DEZIMAL) ";EA
90 PRINTCHR$(147)CHR$(17)CHR$(17)CHR$(17)A/2"REM "F$
100 PRINTCHR$(17)CHR$(17)A/2+1"AA="AA":EA="EA":A="A:PRINTCHR$(17)A":
110 :
120 Z=AA:REM ZAEHLER
130 PRINT"A="A"+1:AZ="AZ":Z="Z":EA="EA:REM DATENUEBERGABE AUF SCHIRM
140 PRINTCHR$(17)CHR$(17)"GOTO150":GOTO260
150 DIMA$(AZ):PS=0:FORI=0TOAZ:IFEA=>ZTHENY=PEEK(Z):Z=Z+1:PS=PS+Y:GOTO170
160 MX=AZ:AZ=I:I=MX:GOTO190
170 A$=STR$(Y):A$(I)=MID$(A$,2)
180 IFLEN(A$(I))<3THENA$(I)="0"+A$(I):GOTO180
190 NEXTI:PRINTCHR$(147)
200 PRINTCHR$(19)CHR$(17)CHR$(17)CHR$(17)A"DATA";:FORI=0TOAZ:PRINTA$(I)",";
210 NEXTI
220 :
230 IFZ>EATHENPRINTCHR$(157)"*"+STR$(PS):PRINT"GOTO270":GOTO260
240 PRINT"*"+STR$(PS):PRINT"A="A"+1:AZ="AZ":Z="Z":EA="EA
250 PRINTCHR$(17)CHR$(17)"GOTO150"
260 PRINTCHR$(19):FORI=631TO638:POKEI,13:NEXT:POKE198,8:END
270 A=A+10:PRINTCHR$(147)CHR$(17)CHR$(17)CHR$(17)A:PRINTA+10:PRINTA+20
280 PRINTA+30:PRINTA+40:PRINT"A="A+40
290 IFA<=250THENPRINTCHR$(17)CHR$(17)CHR$(17)"GOTO270"CHR$(19)
300 PRINTCHR$(19):FORI=631TO640:POKEI,13:NEXT:POKE198,10:END
502 FORI=AATOEA+1
503 READD$
504 IFLEFT$(D$,1)=CHR$(42)THENP$=MID$(D$,2):P=VAL(P$):ZZ=ZZ+1
505 IFPTHENIFS<>PTHENPRINT"FEHLER IN ZEILE: "A+ZZ:STOP
506 IFPTHENS=0:I=I+1:IFI=>EATHENPRINT"FERTIG":END
507 IFPTHENP=0:S=0:I=I-2:NEXTI
508 D=VAL(D$):S=S+D
509 PRINTCHR$(147)"ADRESSE  STRING  DATEN  PEEK  ZEILE
510 PRINTITAB(9)D$TAB(16)D;:POKEI,D:PRINTTAB(23)PEEK(I)TAB(29)ZZ+A+1:NEXTI


READY.



STANDARD-DATA-GENERATOR
```

```
10 REM DEFINITIONEN UND FESTLEGUNGEN
20 A=1000:AZ=7
30 PRINTCHR$(147)"COMPUTERSCHAU HEX-STANDARD-DATAGENERATOR":PRINT"FUER C 64
40  INPUT"FILE-NAME";F$
50  INPUT"ANFANGSADRESSE (VIERSTELLIG $) ";X$:AA$=X$:GOSUB600:AA=X
60  INPUT"ENDADRESSE     (VIERSTELLIG $) ";X$:EA$=X$:GOSUB600:EA=X
70 PRINTCHR$(147)CHR$(17)CHR$(17)CHR$(17)A/2"REM "F$" "AA$" BIS "EA$
80 PRINTCHR$(17)CHR$(17)A/2+1"AA="AA":EA="EA":A="A:PRINTCHR$(17)A":
90 Z=AA:REM ZAEHLER
100  PRINT"A="A"+1:AZ="AZ":Z="Z":EA="EA:REM DATENUEBERGABE AUF SCHIRM
110  PRINTCHR$(17)CHR$(17)"GOTO120":GOTO220:REM 1
120  DIMA$(AZ):X=Z:GOSUB250:DU$="$"+X$+",":PS=0:FORI=0TOAZ:REM 1
130  IFEA=>ZTHENY=PEEK(Z):Z=Z+1:PS=PS+Y:GOTO150
140  MX=AZ:AZ=I:I=MX:GOTO160
150  X=Y:GOSUB250:A$=X$:A$(I)=A$
160  NEXTI:PRINTCHR$(147)
170  PRINTCHR$(19)CHR$(17)CHR$(17)CHR$(17)A"DATA"DU$;:FORI=0TOAZ:PRINTA$(I)",";
180  NEXTI
190  IFZ>EATHENX=PS:GOSUB250:PRINTCHR$(157)"*"+X$:PRINT"GOTO310":GOTO220:REM 2
200 X=PS:GOSUB250:PRINT"*"+X$:PRINT"A="A"+1:AZ="AZ":Z="Z":EA="EA
210  PRINTCHR$(17)CHR$(17)"GOTO120":REM 1
220  PRINTCHR$(19):FORI=631TO638:POKEI,13:NEXT:POKE198,8:END
230  :REM END
240  REM UMWANDLUNG DEZ -> HEX
250  X$="":FORJ=1TO4:X0=X/16:X=X-INT(X0)*16:X$=CHR$(48+X-(X=>10)*7)+X$:X=X0
260  NEXTJ
270  IFLEN(X$)>2THENIFLEFT$(X$,1)="0"THENX$=MID$(X$,2):GOTO270
280  IFLEN(X$)<2THENX$="0"+X$:GOTO280
290  RETURN
300  :
310  A=A+10:PRINTCHR$(147)CHR$(17)CHR$(17)CHR$(17)A:PRINTA+10:PRINTA+20
320  PRINTA+30:PRINTA+40:PRINT"A="A+40
330  IFA<=300THENPRINTCHR$(17)CHR$(17)CHR$(17)"GOTO310"CHR$(19)
340  PRINTCHR$(19):FORI=631TO638:POKEI,13:NEXT:POKE198,8:END
550  FORI=AATOEA+1:READD$
560  IFLEFT$(D$,1)="$"THENX$=MID$(D$,2):I=I-1:NEXT
570  IFLEFT$(D$,1)="*"THENX$=MID$(D$,2):GOSUB600:P=X:ZZ=ZZ+1:I=I+1
580  IFPTHENIFS<>PTHENPRINT"FEHLER IN ZEILE "A+ZZ:STOP
581  IFPTHENS=0:IFI>EA+1THENPRINT"FERTIG":END
582  IFPTHENP=0:S=0:I=I-2:NEXTI
583  X$=D$:GOSUB600:D=X:S=S+D
584  PRINTCHR$(147)"ADRESSE  STRING  DATEN  PEEK  ZEILE
585  PRINTITAB(9)D$TAB(16)D;:POKEI,D:PRINTTAB(23)PEEK(I)TAB(29)ZZ+A+1:NEXTI:END
600  REM UMWANDLUNG HEX -> DEZ (X$ -> X)
610  X=0:FORJ=1TOLEN(X$):X0=ASC(MID$(X$,J,1)):X=16*X+X0-48+(X0>64)*7:NEXTJ:RETURN


READY.



HEX-STANDARD-DATA-GENERATOR
```


Der ComputerSchau-Maschinensprache-Monitor (Com Mon 64)

Um Maschinen- oder auch Basicprogramme im Arbeitsspeicher des C64 oder auch das Betriebssystem des Computers näher kennenzulernen und zu verstehen, ist ein MSM (Maschinen-Sprache-Monitor) ein Hilfsmittel unter verschiedenen anderen, wie z.B. Reassembler, ROM-Listing, Diskettenmonitor, Handbüchern usw.

Auch die Geschwister des Commodore 64 sind in dieser Hinsicht nicht gerade perfekt ausgerüstet. Zwar haben die anderen, älteren „Commodores" den TIM-Monitor resistent im ROM, aber dieser ist nicht gerade leistungsfähig. Er ermöglicht zwar einen Einblick in die Speicherzellen der entsprechenden Computer, aber er besitzt weder einen Assembler- noch einen Disassemblerteil, und beides ist unbedingt erforderlich, wenn man sich etwas näher mit der Maschinensprache beschäftigen will.
Ein Autor dieses Sonderheftes hat bereits einen Disassembler für die alten-Pet-Computer geschrieben, weil es zum damaligen Zeitpunkt noch keine entsprechenden Programme in Deutschland gab. Die Schwierigkeit damals war, daß der PET sich durch „Peeken" in die ROM-Speicher nicht bereit erklären wollte, Informationen „herauszurücken". Ein normales peek (ROM-Bereich) ergab prinzipiell immer eine Null als Antwort. Damals war wahrscheinlich der Schutz des Betriebssystemes mit ein Grund, weshalb man dies seitens des Herstellers nicht zuließ. Ein Ausweg war, diese Abfrage in Maschinensprache durchzuführen. Der Autor hat dann dieses Programm in einer der damaligen Fachzeitschriften veröffentlicht und wahrscheinlich damit einen, wenn auch kleinen, Beitrag zum Verständnis der Commodore-Computer beigetragen. Dokumentierte ROM-Listings erschienen erst später.

Heute und auch bereits ab der CBM 3000-er Serie sieht es schon positiver aus und deswegen ist es auch für viele Anwender leichter, Zusatzprogramme in Maschinensprache zu schreiben. Leider hat Commodore bei der Konzeption des C64 wieder einen Schritt zurück getan, zumindest was den TIM betrifft, er wurde – zugunsten welcher Fähigkeiten auch immer – wieder weggelassen. Andererseits aber gibt es natürlich die Möglichkeit, derartige Programme als Software oder als Zusatzsteckmodul zu erwerben. Die Leistungsfähigkeit der heute erhältlichen MSM geht weit über das hinaus, was der TIM ermöglichte. Aufgrund der wirklich hohen Zahl an erhältlichen MSM haben wir uns lange überlegt, ob es sinnvoll ist, in unserem Sonderheft ein Programm dieser Art aufzunehmen. Die Entscheidung fiel dann zugunsten der Aufnahme und zwar aus nachfolgenden Gründen.

Nicht jeder Leser will unbedingt wirklich in Maschinensprache programmieren, sondern vielleicht nur etwas „hineinschmecken" in seine Maschine, seinen C64. Einen Rat, welchen MSM er hierfür einsetzen soll, wollten wir nicht geben, denn die verschiedenen MSM sind in ihrer Leistungsfähigkeit doch recht unterschiedlich und außerdem kosten sie auch Geld, manchmal sogar zuviel davon. Ein weiterer Punkt, der uns zu unserer Entscheidung „verhalf" war, daß wir getreu unserem Motto: „ComputerSchau zeigt, wie man's macht" eine Grundversion eines derartigen MSM schreiben wollten, der dann von interessierten Lesern auch noch ausbaubar sein sollte. Dadurch, daß wir dann auch noch zwei in unseren Augen vernünftige Standard-Data-Generatoren mit dem dazugehörigen Poke-Generator geschrieben hatten, fanden wir einen Weg, die Eingabe eines derartigen „Programmschlauches" sinnvoll zu „überwachen".
Die Basis zu unserem Com Mon 64 bildeten der TIM und der Supermonitor. Gerade deshalb, weil wir den TIM als Grundlage nahmen, kann jeder, der sich dafür interessiert, unter Zuhilfenahme der vorhandenen dokumentierten ROM-Listings, Änderungen und Erweiterungen in dieses Programm einbauen. Viele Passagen stimmen mit dem TIM vollkommen überein.

Dies alles zu den Vorbemerkungen.

„COM MON 64"

Doch nun zum Programm selbst. Der Com Mon 64 liegt im Adreßbereich $c000...$c908, also in einem Bereich, welcher von Basic aus normalerweise (außer durch peek und poke) nicht benützt wird.

Der Aufruf erfolgt durch Sys12★4096.
An Menü-Programmen stehen folgende Routinen (diese werden auch beim sog. Call-Entry, also dem Aufruf über Sys 12★4096 als Befehlsaufstellung ausgegeben), zur Verfügung:

a ssembliere
d isassembliere
f ill
g oto
h unt
l oad
m emory dump
r egisteranzeige
s ave
t ransfer
x ausstieg aus dem Monitor
← disassembliere von...bis
↑ hardcopy

Nun zu den Befehlen im einzelnen.

Wichtig ist in jedem Falle die Syntax, d. h. die richtige Eingabe, damit das Programm auch das tut, was man will. Um die Handhabung besser zu verstehen, bedienen wir uns immer kleiner Beispiele, welche von Ihnen dann direkt ausgeführt werden können.

Assembliere

Dieser Programmteil ist ein sogenannter Line by Line- oder Direktassembler.
Geben Sie bitte ein:
.a 033c lda #$43 (RETURN)
nach dieser Eingabe müßte der blinkende Cursor unterhalb Ihrer Eingabe-Zeile unter dem „l" von lda stehen, links davon sollte am Bildschirm zu sehen sein: „.a 033e". Falls dies nicht der Fall ist, sollten Sie nochmals mit der ersten Eingabe beginnen. Drücken Sie zu diesem Zwecke einfach mehrmals die Return-Taste und beginnen die Eingabe nochmals neu.
Fahren Sie fort mit:
sta $0400
rts
Nach dem rts drücken Sie bitte die Return-Taste, danach x, dann Return. Ihr Bildschirm sollte nun in etwa folgendes Bild zeigen:

```
.A 033C LDA #$43
.A 033E STA $0400
.A 0341 RTS
.A 0342
.X
READY.
```

Nun zur Erklärung, was Sie getan haben. Sie haben ein kleines Maschinenprogramm geschrieben, welches beim Start den Buchstaben „C" in die linke obere Ecke des Bildschirmes schreibt.
Doch nun wieder erst der Reihe nach.
Die erste Eingabezeile bewirkte, daß das Programm in den Assembliermodus ging, außerdem sollte der Akkumulator absolut, das heißt mit der Zahl selbst und nicht mit dem Speicherinhalt der Adresse $43 geladen werden. Da das Programm erkannt hatte, daß assembliert werden soll, wurde die nächste Assembleradresse, nämlich $033e ausgegeben und der Cursor bereits an die für die Eingabe richtige Position gesetzt. Der Befehl sta $0400 bewirkt im später aufgerufenen Programm dann, daß das Zeichen, welches sich im Akku befindet, in die erste Bildschirmadresse geschrieben wird. Der nächste Befehl, nämlich rts ist die Abkürzung von Return SubRoutine und macht nichts anderes wie der Befehl „return" in Basic: also Rücksprung zu der aufrufenden Routine. Das Drücken der Return-Taste bewirkte dann, daß aus dem Assembliermodus „ausgestiegen" wurde, und das „x" mit anschließendem Return bewirkte den Rücksprung aus dem MSM in den normalen Basicmodus. Wir wollen nun gleich unser kleines Maschinenprogramm starten. Geben Sie deswegen bitte ein:
Sys 828
Nun sollte in der linken oberen Bildschirmecke, wie bereits schon weiter oben geschrieben, der Großbuchstabe „C" erscheinen. Falls aber, weil Sie sich in der letzten Bildschirmzeile befanden, ein „Bildschirmscrolling" also ein Hochschieben der Zeilen erfolgte, dann fahren Sie nun einfach mit dem Cursor wieder auf den Sys-Befehl, Return... und nun müßte das vorausgesagte Ergebnis zu sehen sein.
Um nun wieder in den MSM-Modus zu kommen, geben Sie bitte ein: Sys12★4096.
Es geht auch anders, denn Sie brauchen lediglich eine Adresse anzuspringen, welche eine Null enthält. Denn der Break-Vektor wurde beim erstmaligen Aufruf auf den MSM gestellt, und zwar auf die Adresse $c02c, dem eigentlichen Beginn des ComMon 64! Mit Sys 2047 müßte es eigentlich immer klappen.

Disassembliere

Wenn Sie wieder im MSM-Betrieb sind, geben Sie nun bitte ein:
d 033c (Return)
Wie Sie leicht erkennen können, wird nun genau das Programm, welches Sie erst vorhin eingegeben haben, disassembliert und zwar immer eine Bildschirmseite voll. Um nun weitere Disassemblierungen vorzunehmen, brauchen Sie nur d und return zu drücken, schon erscheint die Fortsetzung. Wollen Sie an einer anderen Adresse weitermachen, so geben Sie ein:
d xxxx (xxxx = Hexadezimale Adresse des Disassemblierstartes)
Diese Routine dürfte Ihnen keine größeren Probleme bereiten, deswegen weiter mit dem nächsten Befehl.

Fill

Syntax: f aaaa eeee zz
hierbei bedeuten
f = Fülle
aaaa = Anfangsadresse hexadezimal
eeee = Endadresse hexadezimal
zz = Zeichen
Dieser Befehl bewirkt, daß ab der angegebenen Anfangs- (bis zur Endadresse) alle Speicherstellen mit dem Zeichen zz, welches Sie frei wählen können, gefüllt wird.
Dabei sind natürlich nur Hexzahlen im Bereich von „00" bis „ff" möglich.
Beispiel:
f 033c,0350,ff oder f 033c 0350 ff
Wenn Sie nun disassemblieren, können Sie sehen, daß der angegebene Bereich tatsächlich mit ihrem definierten Zeichen gefüllt wurde. (Sie können es natürlich auch durch einen „Memory-dump" feststellen.)
Aufgabe dieser Routine ist es, einen bestimmten Speicherbereich mit bestimmten Zeichen (sehr oft „00", „ea" oder „ff") zu füllen. Warum gerade mit diesen Zeichen? Da kommen Sie sicher ganz von alleine darauf.

Goto

Syntax: g aaaa
aaaa = wieder die Adresse in Hexform.
Dies bewirkt, daß ein Programm (Maschinenprogramm) ab der angegebenen Adresse abgearbeitet wird.
Falls Sie noch das vorhin geschriebene kleine Maschinenprogramm im Speicher haben, können Sie ohne Probleme mittels „g 033c" den Buchstaben „c" in die l.. o.. B.. usw. ausgeben lassen. Wir hoffen, Sie kamen mit den Abkürzungen eben klar. Falls nein, kurze Rückfrage unsererseits: Wo wird das Zeichen ausgegeben? Aha!
Den gleichen Effekt können Sie auch erreichen, indem Sie durch den Befehl R die Register ausgeben lassen und den PC (Programm-Counter = Programmzähler) auf die entsprechende Adresse stellen und anschließend nur durch die Eingabe von „g" diese Routine aufrufen.

```
500 REM COM MON 64.DEZDATA
501 AA= 49152 :EA= 51464 :A= 1000
502 FORI=AATOEA+1
503 READD$
504 IFLEFT$(D$,1)=CHR$(42)THENP$=MID$(D$,2):P=VAL(P$):ZZ=ZZ+1
505 IFPTHENIFS<>PTHENPRINT"FEHLER IN ZEILE: "A+ZZ:STOP
506 IFPTHENS=0:I=I+1:IFI=>EATHENPRINT"FERTIG":END
507 IFPTHENP=0:S=0:I=I-2:NEXTI
508 D=VAL(D$):S=S+D
509 PRINTCHR$(147)"ADRESSE  STRING  DATEN  PEEK  ZEILE
510 PRINTITAB(9)D$TAB(16)D:POKEI,D:PRINTTAB(23)PEEK(I)TAB(29)ZZ+A+1:NEXTI
1000 :
1001 DATA169,000,141,033,208,141,032,208,173,058,200,201,000,240,009,* 1813
1002 DATA032,210,255,238,009,192,076,008,192,169,058,141,009,192,169,* 1950
1003 DATA104,141,022,003,169,192,141,023,003,169,128,032,144,255,000,* 1526
1004 DATA000,000,000,000,058,059,082,077,071,088,076,083,084,070,072,* 820
1005 DATA068,095,044,065,094,000,000,000,171,193,158,193,053,193,096,* 1423
1006 DATA193,191,193,242,193,093,194,117,194,167,195,251,195,041,196,* 2655
1007 DATA161,196,191,197,238,197,016,198,139,200,000,000,000,000,216,* 1949
1008 DATA104,141,062,002,104,141,061,002,104,141,060,002,104,141,059,* 1228
1009 DATA002,104,170,104,168,056,138,233,002,141,058,002,152,233,000,* 1563
1010 DATA141,057,002,186,142,063,002,032,187,197,162,066,169,042,032,* 1480
1011 DATA192,194,169,082,208,052,230,193,208,006,230,194,208,002,230,* 2398
1012 DATA038,096,032,207,255,201,013,208,248,104,104,169,005,032,210,* 1922
1013 DATA255,169,000,133,038,162,013,169,046,032,192,194,169,005,032,* 1609
1014 DATA210,255,032,167,192,201,046,240,249,201,032,240,245,162,015,* 2487
1015 DATA221,049,192,208,012,138,010,170,189,069,192,072,189,068,192,* 1971
1016 DATA072,096,202,016,236,076,086,195,165,193,141,058,002,165,194,* 1897
1017 DATA141,057,002,096,169,008,133,029,160,000,032,184,197,177,193,* 1578
1018 DATA032,177,194,032,156,192,198,029,208,241,096,032,241,194,144,* 2166
1019 DATA011,162,000,129,193,193,193,240,003,076,086,195,032,156,192,* 1861
1020 DATA198,029,096,169,059,133,193,169,002,133,194,169,005,096,152,* 1797
1021 DATA072,032,187,197,104,162,046,076,192,194,169,005,032,210,255,* 1933
1022 DATA162,000,189,031,200,032,210,255,232,224,022,208,245,160,059,* 2229
1023 DATA032,043,193,173,057,002,032,177,194,173,058,002,032,177,194,* 1539
1024 DATA032,032,193,032,246,192,240,092,032,167,192,032,226,194,144,* 2046
1025 DATA051,032,210,194,032,167,192,032,226,194,144,040,032,210,194,* 1950
1026 DATA169,005,032,210,255,032,225,255,240,060,166,038,208,056,165,* 2116
1027 DATA195,197,193,165,196,229,194,144,046,160,058,032,043,193,032,* 2077
1028 DATA170,194,032,244,192,240,224,076,086,195,032,226,194,144,003,* 2252
1029 DATA032,233,192,032,032,193,208,007,032,226,194,144,235,169,008,* 1937
1030 DATA133,029,032,167,192,032,010,193,208,248,076,176,192,032,207,* 1927
```

```
1031 DATA255,201,013,240,012,201,032,208,209,032,226,194,144,003,032,* 2002
1032 DATA233,192,169,005,032,210,255,174,063,002,154,120,173,057,002,* 1841
1033 DATA072,173,058,002,072,173,059,002,072,173,060,002,174,061,002,* 1155
1034 DATA172,062,002,064,169,005,032,210,255,174,063,002,154,076,123,* 1563
1035 DATA227,160,001,132,186,132,185,136,132,183,132,144,132,147,169,* 2198
1036 DATA064,133,187,169,002,133,188,032,207,255,201,032,240,249,201,* 2293
1037 DATA013,240,056,201,034,208,020,032,207,255,201,034,240,016,201,* 1958
1038 DATA013,240,041,145,187,230,183,200,192,016,208,236,076,086,195,* 2248
1039 DATA032,207,255,201,013,240,022,201,044,208,220,032,241,194,041,* 2151
1040 DATA015,240,233,201,003,240,229,133,186,032,207,255,201,013,096,* 2284
1041 DATA108,048,003,108,050,003,032,255,193,208,212,169,005,032,210,* 1636
1042 DATA255,169,000,032,088,194,165,144,041,016,208,196,076,176,192,* 1952
1043 DATA032,255,193,201,044,208,186,032,226,194,032,210,194,032,207,* 2246
1044 DATA255,201,044,208,173,032,226,194,165,193,133,174,165,194,133,* 2490
1045 DATA175,032,210,194,032,207,255,201,013,208,152,169,005,032,210,* 2095
1046 DATA255,032,091,194,076,176,192,165,194,032,177,194,165,193,072,* 2208
1047 DATA074,074,074,074,032,201,194,170,104,041,015,032,201,194,072,* 1552
1048 DATA138,032,210,255,104,076,210,255,009,048,201,058,144,002,105,* 1847
1049 DATA006,096,162,002,181,192,072,181,194,149,192,104,149,194,202,* 2076
1050 DATA208,243,096,032,241,194,144,002,133,194,032,241,194,144,002,* 2100
1051 DATA133,193,096,169,000,133,042,032,167,192,201,032,208,009,032,* 1639
1052 DATA167,192,201,032,208,014,024,096,032,024,195,010,010,010,010,* 1225
1053 DATA133,042,032,167,192,032,024,195,005,042,056,096,201,058,144,* 1419
1054 DATA002,105,008,041,015,096,162,002,044,162,000,180,193,208,008,* 1226
1055 DATA180,194,208,002,230,038,214,194,214,193,096,032,167,192,201,* 2355
1056 DATA032,240,249,096,169,000,141,000,001,032,053,195,032,248,194,* 1682
1057 DATA032,229,194,144,009,096,032,167,192,032,226,194,176,222,174,* 2119
1058 DATA063,002,154,169,005,032,210,255,169,063,032,210,255,076,176,* 1871
1059 DATA192,032,184,197,202,208,250,096,230,195,208,002,230,196,096,* 2518
1060 DATA162,002,181,192,072,181,039,149,192,104,149,039,202,208,243,* 2115
1061 DATA096,165,195,164,196,056,233,002,176,014,136,144,011,165,040,* 1793
1062 DATA164,041,076,156,195,165,195,164,196,056,229,193,133,030,152,* 2145
1063 DATA229,194,168,005,030,096,032,061,195,032,210,194,032,078,195,* 1751
1064 DATA032,117,195,032,078,195,032,152,195,032,210,194,144,021,166,* 1795
1065 DATA038,208,100,032,145,195,144,095,161,193,129,195,032,110,195,* 1972
1066 DATA032,156,192,208,235,032,145,195,024,165,030,101,195,133,195,* 2038
1067 DATA152,101,196,133,196,032,117,195,166,038,208,061,161,193,129,* 2078
1068 DATA195,032,145,195,176,052,032,033,195,032,036,195,076,230,195,* 1819
1069 DATA032,061,195,032,210,194,032,078,195,032,210,194,032,167,192,* 1856
1070 DATA032,241,194,144,020,133,029,166,038,208,017,032,152,195,144,* 1745
1071 DATA012,165,029,129,193,032,156,192,208,238,076,086,195,076,176,* 1963
1072 DATA192,032,061,195,032,210,194,032,078,195,032,210,194,032,167,* 1856
1073 DATA192,162,000,032,167,192,201,039,208,020,032,167,192,157,016,* 1777
1074 DATA002,232,032,207,255,201,013,240,034,224,032,208,241,240,028,* 2189
1075 DATA142,000,001,032,248,194,144,198,157,016,002,232,032,207,255,* 1860
1076 DATA201,013,240,009,032,241,194,144,182,224,032,208,236,134,028,* 2118
1077 DATA169,005,032,210,255,032,187,197,162,000,160,000,177,193,221,* 2000
1078 DATA016,002,208,012,200,232,228,028,208,243,032,170,194,032,184,* 1989
1079 DATA197,032,156,192,166,038,208,141,032,152,195,176,221,076,176,* 2158
1080 DATA192,032,061,195,133,032,165,194,133,033,162,000,134,040,169,* 1675
1081 DATA147,032,210,255,169,024,133,029,032,206,196,032,046,197,133,* 1841
1082 DATA193,132,194,198,029,208,242,169,145,032,210,255,076,176,192,* 2451
1083 DATA160,044,032,043,193,032,184,197,032,170,194,032,184,197,162,* 1856
1084 DATA000,161,193,032,061,197,072,032,131,197,104,032,153,197,162,* 1724
1085 DATA006,224,003,208,018,164,031,240,014,165,042,201,232,177,193,* 1918
1086 DATA176,028,032,038,197,136,208,242,006,042,144,014,189,145,199,* 1796
1087 DATA032,009,198,189,151,199,240,003,032,009,198,202,208,213,096,* 1979
1088 DATA032,049,197,170,232,208,001,200,152,032,038,197,138,134,028,* 1808
1089 DATA032,177,194,166,028,096,165,031,056,164,194,170,016,001,136,* 1626
1090 DATA101,193,144,001,200,096,168,074,144,011,074,176,023,201,034,* 1640
1091 DATA240,019,041,007,009,128,074,170,189,061,199,176,004,074,074,* 1465
1092 DATA074,074,041,015,208,004,160,128,169,000,170,189,132,199,133,* 1696
1093 DATA042,041,003,133,031,152,041,143,170,152,160,003,224,138,240,* 1673
1094 DATA011,074,144,008,074,074,009,032,136,208,250,200,136,208,242,* 1806
```



```
1095 DATA096,177,193,032,038,197,162,001,032,103,195,196,031,200,144,* 1797
1096 DATA241,162,003,192,004,144,242,096,168,185,158,199,133,040,185,* 2152
1097 DATA222,199,133,041,169,000,160,005,006,041,038,040,042,136,208,* 1440
1098 DATA248,105,063,032,210,255,202,208,236,169,032,044,169,013,076,* 2062
1099 DATA210,255,032,061,195,032,210,194,032,078,195,032,210,194,162,* 2092
1100 DATA000,134,040,169,005,032,210,255,032,187,197,032,214,196,032,* 1735
1101 DATA046,197,133,193,132,194,032,225,255,240,005,032,152,195,176,* 2207
1102 DATA233,076,176,192,032,061,195,169,003,133,029,032,167,192,032,* 1722
1103 DATA010,193,208,248,165,032,133,193,165,033,133,194,076,175,196,* 2154
1104 DATA197,040,240,003,032,210,255,096,032,061,195,032,210,194,142,* 1939
1105 DATA017,002,162,003,032,053,195,072,202,208,249,162,003,104,056,* 1520
1106 DATA233,063,160,005,074,110,017,002,110,016,002,136,208,246,202,* 1584
1107 DATA208,237,162,002,032,207,255,201,013,240,030,201,032,240,245,* 2305
1108 DATA032,052,199,176,015,032,005,195,164,193,132,194,133,193,169,* 1884
1109 DATA048,157,016,002,232,157,016,002,232,208,219,134,040,162,000,* 1625
1110 DATA134,038,240,004,230,038,240,117,162,000,134,029,165,038,032,* 1601
1111 DATA061,197,166,042,134,041,170,188,158,199,189,222,199,032,029,* 2027
1112 DATA199,208,227,162,006,224,003,208,025,164,031,240,021,165,042,* 1925
1113 DATA201,232,169,048,176,033,032,035,199,208,204,032,037,199,208,* 2013
1114 DATA199,136,208,235,006,042,144,011,188,151,199,189,145,199,032,* 2084
1115 DATA029,199,208,181,202,208,209,240,010,032,028,199,208,171,032,* 2156
1116 DATA028,199,208,166,165,040,197,029,208,160,032,210,194,164,031,* 2031
1117 DATA240,040,165,041,201,157,208,026,032,133,195,144,010,152,208,* 1952
1118 DATA004,165,030,016,010,076,086,195,200,208,250,165,030,016,246,* 1697
1119 DATA164,031,208,003,185,194,000,145,193,136,208,248,165,038,145,* 2063
1120 DATA193,032,046,197,133,193,132,194,169,005,032,210,255,160,065,* 2016
1121 DATA032,043,193,032,184,197,032,170,194,032,184,197,169,005,032,* 1696
1122 DATA210,255,076,020,198,168,032,035,199,208,017,152,240,014,134,* 1958
1123 DATA028,166,029,221,016,002,008,232,134,029,166,028,040,096,201,* 1396
1124 DATA048,144,003,201,071,096,056,096,064,002,069,003,208,008,064,* 1133
1125 DATA169,019,120,169,000,033,129,130,000,000,089,077,145,146,134,* 1360
1126 DATA064,009,064,002,069,179,208,008,064,009,000,034,068,051,208,* 1037
1127 DATA140,068,000,017,034,068,051,208,140,068,154,016,034,068,051,* 1117
1128 DATA208,008,064,009,016,034,068,051,208,008,064,009,098,019,120,* 984
1129 DATA169,019,120,169,000,033,129,130,000,000,089,077,145,146,134,* 1360
1130 DATA074,133,157,044,041,044,035,040,036,089,000,088,036,036,000,* 853
1131 DATA028,138,028,035,093,139,027,161,157,138,029,035,157,139,029,* 1333
1132 DATA161,000,041,025,174,105,168,025,035,036,083,027,035,036,083,* 1034
1133 DATA025,161,000,026,091,091,165,105,036,036,174,174,168,173,041,* 1466
1134 DATA000,124,000,021,156,109,156,165,105,041,083,132,019,052,017,* 1180
1135 DATA165,105,035,160,216,098,090,072,038,098,148,136,084,068,200,* 1713
1136 DATA084,104,068,232,148,000,180,008,132,116,180,040,110,116,244,* 1762
1137 DATA204,074,114,242,164,138,000,170,162,162,116,116,116,114,068,* 1960
1138 DATA104,178,050,178,000,034,000,026,026,038,038,114,114,136,200,* 1236
1139 DATA196,202,038,072,068,068,162,200,000,013,032,032,032,080,067,* 1262
1140 DATA032,032,083,082,032,065,067,032,088,082,032,089,082,032,083,* 913
1141 DATA080,000,000,000,000,000,147,018,005,195,079,077,080,085,084,* 850
1142 DATA069,082,211,067,072,065,085,045,205,079,078,073,084,079,082,* 1376
1143 DATA032,032,032,032,032,032,040,067,065,076,076,045,069,078,084,* 792
1144 DATA082,089,041,032,013,194,069,070,069,072,076,069,058,032,065,* 1031
1145 DATA032,068,032,070,032,071,032,072,032,076,032,077,032,082,032,* 772
1146 DATA083,032,084,032,088,032,095,032,094,014,013,000,000,169,127,* 895
1147 DATA162,004,160,000,032,186,255,169,000,032,189,255,032,192,255,* 1923
1148 DATA176,088,169,000,133,253,169,004,133,254,162,127,032,201,255,* 2156
1149 DATA162,025,169,013,032,210,255,169,029,160,005,234,032,210,255,* 1960
1150 DATA136,208,250,032,210,255,032,225,255,240,049,160,000,177,253,* 2482
1151 DATA133,252,041,063,006,252,036,252,016,002,009,128,112,002,009,* 1313
1152 DATA064,032,210,255,200,192,040,208,230,152,024,101,253,133,253,* 2347
1153 DATA144,002,230,254,202,208,191,169,013,032,210,255,032,204,255,* 2401
1154 DATA169,127,032,195,255,169,157,032,210,255,169,044,032,210,255,* 2311
1155 DATA076,176,192,* 444

READY.
```

Der ComputerSchau-Maschinensprache-Monitor (Com Mon 64)

(Teil 2)

Hunt (= Suche)

Bei angegebener Anfangs- und Endadresse wird durch diesen Befehl der Speicherbereich nach einer Bytefolge oder nach den ASCII-Zeichen durchsucht und die Adressen auf den Bildschirm ausgegeben, die diese Werte beinhalten.
Beispiele für die Syntax:
h c000 cfff h1 h2 h3 h4
oder: h c000,cfff,h1,h2,h3,h4
oder: h c000 cfff 'ASCII-Folge
oder: h c000,cfff,'ASCII-Folge
(h1...h4 stehen für 2stellige Hexzahlen)
Anwendungsbeispiel:
h a000 ffff 'bytes free
Antwort: e467
Dort genau steht also die Meldung, die der C64 beim Einschalten bringt!
Spielen Sie ruhig etwas mit dieser Funktion und finden Sie heraus, wo verschiedene ASCII-Folgen stehen.

Load

Ebenfalls wie in Basic, ist dies der Ladebefehl für Programme. Allerdings ist die Syntax geringfügig unterschiedlich, sie lautet:
l "Programm-Name",ga
ga = die Geräteadresse, also für den Kassettenrecorder 01, für die Floppy-Station normalerweise 08! Die Eingabe muß immer mit Komma nach dem Programm-Namen und die Geräte-Adresse zweistellig erfolgen.
Wichtig zu wissen ist, daß aus dem Monitor heraus immer an die Originaladresse, welche sich entweder im „Header" der Kassette oder auf der Diskette befindet, geladen wird.

Der nächste Befehl ist:

Memory Dump (Speicherausgabe in hexadezimaler Darstellung)

Syntax:
m aaaa eeee oder m aaaa,eeee
Die Bezeichnungen dürften nun bereits geläufig sein. Wenn nicht, dann bitte im zuvor „Geschriebenen" nachsehen.
Beispiel: siehe Hardcopy „Memory-Dump"

Sie können natürlich auch die Speicherstellen mit neuen Werten überschreiben. Hierzu brauchen Sie nur mit dem Cursor auf die entsprechenden Stellen zu fahren und zu ändern.

Register-Display

Diese Routine zeigt die Speicherinhalte der verschiedenen Register an. Diese sind:
pc (Programm-Zähler = Programm-Counter)
sr (Status-Register)
ac (Akkumulator)
xr (x-Register)
yr (y-Register)
sp (Stapelzeiger = Stack-Pointer)

```
M C000 C050
.:C000 A9 00 8D 21 D0 8D 20 D0
.:C008 AD 3A C8 C9 00 F0 09 20
.:C010 D2 FF EE 09 C0 4C 08 C0
.:C018 A9 3A 8D 09 C0 A9 68 8D
.:C020 16 03 A9 C0 8D 17 03 A9
.:C028 80 20 90 FF 00 00 00 00
.:C030 00 3A 3B 52 4D 47 58 4C
.:C038 53 54 46 48 44 5F 2C 41
.:C040 5E 00 00 00 AB C1 9E C1
.:C048 35 C1 60 C1 BF C1 F2 C1
.:C050 5D C2 75 C2 A7 C3 FB C3
.
 HARDCOPY: "MEMORY-DUMP"
```


Durch „Überschreiben" können die Werte dieser Speicher geändert werden. Geben Sie bitte beim Programm-Zähler (pc) ein: c000 und drücken Sie dann bitte die Return-Taste. Anschließend geben Sie bitte „g" ein und drücken dann wieder Return. Wie Sie sehen konnten, startete der MSM wieder, genauso als ob Sie von Basic aus Sys12★4096 eingegeben hätten.
Wir hoffen, daß diese kurze Beschreibung für Sie ausreicht.

Save

Wie bei Basic wird dadurch bewirkt, daß ein Programm (in diesem Falle genauer gesagt ein Speicherbereich!!) auf das Gerät mit der Geräteadresse „ga" abgesaved wird. Allerdings gibt es einen ganz gravierenden Unterschied zum „Basic-Save": Der Bereich, welcher gesaved werden soll, muß angegeben werden, ist aber frei wählbar.
Syntax: s „File-Name",ga,aaaa,eeee
Praktisches Beispiel: Der Com Mon 64, welchen Sie beispielsweise über den „Poke-Generator" erzeugt haben, kann direkt abgespeichert werden, wodurch sich die Ladezeit um einiges verkürzt.
Bitte eingeben:
s "com mon 64",08,c000,c909
(Sehr wichtig ist, daß bei dieser Funktion als Endadresse immer die wirkliche Endadresse + 1 eingegeben werden muß!)
Dadurch wird der MSM auf Diskette geschrieben und kann durch ganz normales
load"com mon 64",08,01 wieder in den C64 eingeladen werden.
Vor dem Aufruf ist aber der Befehl „new" einzugeben. Danach wieder Sys 12★4096 und Sie sind im Monitor.

Transfer

Dient zum Transferieren von Speicherbereichen.
Syntax: t aada eada aane
aada = Anfangsadresse alt
eada = Endadresse alt
aane = Anfangsadresse neu
Durch diesen Befehl können Sie also ganze Speicherbereiche übertragen, wobei, wenn die neuen Adressen nicht im Bereich der alten liegen, im Ausgangsbereich keine Veränderungen stattfinden. Geben Sie bitte ein:
t c000 cfff 9000,
wie Sie mittels der Kommandos m oder d leicht feststellen können, wurden die Speicherinhalte von $c000...cfff in den Bereich mit der Anfangsadresse $9000 übertragen. Allerdings ist dort der Monitor nicht „lauffähig" da seine Sprungadressen nicht stimmen. Ein Aufruf mit Sys9★4096 würde mit Sicherheit zum „Absturz" führen, wenn der Original-Monitor nicht mehr im Speicher steht.

x = Ausstieg

Der Befehl „x" bewirkt einen Ausstieg aus dem Monitor und damit eine Rückkehr in den normalen Basic-Betrieb.

← = Disassembliere Bereich

Der Befehl „Pfeil nach links" mit nachfolgender Anfangs- und Endadresse bewirkt eine Disassemblierung von der Anfangs- bis zur Endadresse. Der Unterschied zum Befehl „d" ist also, daß Anfangs- und Endadresse im Gegensatz zur „Bildschirm-Seiten-Disassemblierung" gewählt werden können.

↑ = Hardcopy

Der letzte Befehl, aufgerufen durch den Pfeil nach oben, gibt eine Hardcopy des Bildschirminhaltes aus.
Nun noch ein paar Anmerkungen zum Monitorbetrieb bzw. zu ihm selbst. Sie können auch größere Passagen auf einen Drucker ausgeben lassen, hierzu müssen Sie nur ein Druckfile eröffnen, dann CMD und den Monitor aufrufen. Beispiel:
open1,4,7:cmd1:sys12★4096
m c000 c120
Nach der Ausgabe dann Ausstieg mit „x" und das Druckfile wieder schließen.
Falls Sie Ihren Drucker nicht in den Groß-/Kleinschrift-Modus gebracht haben, dies geschieht beim MPS 802 durch die Sekundäradresse 7, dann erscheinen auf dem Papier die Großbuchstaben natürlich als grafische Zeichen, und die „Kleinen" als „Große", Sie wissen doch: ASCII, CBMASCII und Bildschirmcode! Außerdem haben Sie wahrscheinlich bemerkt, daß auch die Startmeldungen nicht mehr auf dem Bildschirm, sondern ebenfalls über den Drucker ausgegeben wurden.

In den Speicherstellen von $c031...$c043 steht der Befehlssatz des MSM, in den letzten drei Bytes steht jeweils eine Null, das heißt, dort können noch weitere zwei (nicht drei) Befehlszeichen eingebracht werden. Die zugehörigen Adressen für die jeweiligen Routinen stehen ab der Adresse $c044 und benötigen immer je zwei Bytes.

Am Ende dieser „Lookup"-Tabelle finden Sie insgesamt viermal die Null, aus diesem Grunde also kann, ohne größeren Aufwand, nur um zwei Zusatzbefehle erweitert werden. Zu berücksichtigen ist, daß diese Sprungadressen immer um 1 Byte niedriger sind als die wirklichen Einsprungpunkte. Die Anzahl der abzufragenden Befehle steht in $c0d1 (Inhalt = 0f also 15).
Durch die Routine ab $c0da werden die beiden Adreßbytes, die bei einem gültigen Befehl über das x-Register geholt werden, auf den Stack gelegt. Mit diesen Angaben sollte es Ihnen gelingen, zumindest kleinere zusätzliche Routinen einzubauen. Wir empfehlen aber, unbedingt ein dokumentiertes ROM-Listing des TIM zu Rate zu ziehen, denn sonst werden Sie sehr lange brauchen, um die Arbeitsweise dieses Programmes zu verstehen.

Wer Spaß an der Programmierung in Maschinensprache findet, sollte sich aber einen komfortableren MSM und auch beispielsweise Label- und Makrofähige Assembler zulegen, denn mit dem im Com Mon 64 implementierten Direktassembler wird es für längere Programmentwicklungen ansonsten doch zu umständlich. Wir hoffen, daß Sie mit dieser Kurzanleitung des Com Mon 64 zurecht kommen. Vor allem aber gilt: „Übung macht den Meister".

Com Mon 64
Maschinensprachemonitor

```
500 REM COM MON DATA HEX C000 BIS C909
501 AA= 49152 :EA= 51465 :A= 1000
550 FORI=AATOEA+1:READD$
560 IFLEFT$(D$,1)="$"THENX$=MID$(D$,2):I=I-1:NEXT
570 IFLEFT$(D$,1)="*"THENX$=MID$(D$,2):GOSUB600:P=X:ZZ=ZZ+1:I=I+1
580 IFPTHENIFS<>PTHENPRINT"FEHLER IN ZEILE "A+ZZ:STOP
581 IFPTHENS=0:IFI>EA+1THENPRINT"FERTIG":END
582 IFPTHENP=0:S=0:I=I-2:NEXTI
583 X$=D$:GOSUB600:D=X:S=S+D
584 PRINTCHR$(147)"ADRESSE  STRING  DATEN  PEEK  ZEILE
585 PRINTITAB(9)D$TAB(16)D;:POKEI,D:PRINTTAB(23)PEEK(I)TAB(29)ZZ+A+1:NEXTI:END
600 REM UMWANDLUNG HEX -> DEZ (X$ -> X)
610 X=0:FORJ=1TOLEN(X$):X0=ASC(MID$(X$,J,1)):X=16*X+X0-48+(X0>64)*7:NEXTJ:RETURN
1000 :
1001 DATA$C000,A9,00,8D,21,D0,8D,20,D0,*3A4
1002 DATA$C008,AD,3A,C8,C9,00,F0,09,20,*391
1003 DATA$C010,D2,FF,EE,09,C0,4C,08,C0,*49C
1004 DATA$C018,A9,3A,8D,09,C0,A9,68,8D,*3D7
1005 DATA$C020,16,03,A9,C0,8D,17,03,A9,*2D2
1006 DATA$C028,80,20,90,FF,00,00,00,00,*22F
1007 DATA$C030,00,3A,3B,52,4D,47,58,4C,*1FF
1008 DATA$C038,53,54,46,48,44,5F,2C,41,*245
1009 DATA$C040,5E,00,00,00,AB,C1,9E,C1,*329
1010 DATA$C048,35,C1,60,C1,BF,C1,F2,C1,*54A
1011 DATA$C050,5D,C2,75,C2,A7,C3,FB,C3,*57E
1012 DATA$C058,29,C4,A1,C4,BF,C5,EE,C5,*589
1013 DATA$C060,10,C6,8B,C8,00,00,00,00,*229
1014 DATA$C068,D8,68,8D,3E,02,68,8D,3D,*33F
1015 DATA$C070,02,68,8D,3C,02,68,8D,3B,*265
1016 DATA$C078,02,68,AA,68,A8,38,8A,E9,*3CF
1017 DATA$C080,02,8D,3A,02,98,E9,00,8D,*2D9
1018 DATA$C088,39,02,BA,8E,3F,02,20,BB,*29F
1019 DATA$C090,C5,A2,42,A9,2A,20,C0,C2,*41E
1020 DATA$C098,A9,52,D0,34,E6,C1,D0,06,*47C
1021 DATA$C0A0,E6,C2,D0,02,E6,26,60,20,*406
1022 DATA$C0A8,CF,FF,C9,0D,D0,F8,68,68,*53C
1023 DATA$C0B0,A9,05,20,D2,FF,A9,00,85,*3CD
1024 DATA$C0B8,26,A2,0D,A9,2E,20,C0,C2,*34E
1025 DATA$C0C0,A9,05,20,D2,FF,20,A7,C0,*426
1026 DATA$C0C8,C9,2E,F0,F9,C9,20,F0,F5,*5AE
1027 DATA$C0D0,A2,0F,DD,31,C0,D0,0C,8A,*3E5
1028 DATA$C0D8,0A,AA,BD,45,C0,48,BD,44,*3BF
1029 DATA$C0E0,C0,48,60,CA,10,EC,4C,56,*3D0
1030 DATA$C0E8,C3,A5,C1,8D,3A,02,A5,C2,*459
1031 DATA$C0F0,8D,39,02,60,A9,08,85,1D,*27B
1032 DATA$C0F8,A0,00,20,B8,C5,B1,C1,20,*3CF
1033 DATA$C100,B1,C2,20,9C,C0,C6,1D,D0,*4A2
1034 DATA$C108,F1,60,20,F1,C2,90,0B,A2,*461
1035 DATA$C110,00,81,C1,C1,C1,F0,03,4C,*403
1036 DATA$C118,56,C3,20,9C,C0,C6,1D,60,*3D8
1037 DATA$C120,A9,3B,85,C1,A9,02,85,C2,*41C
1038 DATA$C128,A9,05,60,98,48,20,BB,C5,*38E
1039 DATA$C130,68,A2,2E,4C,C0,C2,A9,05,*3B4
1040 DATA$C138,20,D2,FF,A2,00,BD,1F,C8,*437
1041 DATA$C140,20,D2,FF,E8,E0,16,D0,F5,*594
1042 DATA$C148,A0,3B,20,2B,C1,AD,39,02,*2CF
1043 DATA$C150,20,B1,C2,AD,3A,02,20,B1,*34D
1044 DATA$C158,C2,20,20,C1,20,F6,C0,F0,*489
1045 DATA$C160,5C,20,A7,C0,20,E2,C2,90,*437
1046 DATA$C168,33,20,D2,C2,20,A7,C0,20,*38E
1047 DATA$C170,E2,C2,90,28,20,D2,C2,A9,*4B9
1048 DATA$C178,05,20,D2,FF,20,E1,FF,F0,*4E6
1049 DATA$C180,3C,A6,26,D0,38,A5,C3,C5,*43D
1050 DATA$C188,C1,A5,C4,E5,C2,90,2E,A0,*52F
1051 DATA$C190,3A,20,2B,C1,20,AA,C2,20,*2F2
1052 DATA$C198,F4,C0,F0,E0,4C,56,C3,20,*509
1053 DATA$C1A0,E2,C2,90,03,20,E9,C0,20,*420
1054 DATA$C1A8,20,C1,D0,07,20,E2,C2,90,*40C
1055 DATA$C1B0,EB,A9,08,85,1D,20,A7,C0,*3C5
1056 DATA$C1B8,20,0A,C1,D0,F8,4C,B0,C0,*46F
1057 DATA$C1C0,20,CF,FF,C9,0D,F0,0C,C9,*489
1058 DATA$C1C8,20,D0,D1,20,E2,C2,90,03,*418
1059 DATA$C1D0,20,E9,C0,A9,05,20,D2,FF,*468
1060 DATA$C1D8,AE,3F,02,9A,78,AD,39,02,*2E9
1061 DATA$C1E0,48,AD,3A,02,48,AD,3B,02,*263
1062 DATA$C1E8,48,AD,3C,02,AE,3D,02,AC,*2CC
1063 DATA$C1F0,3E,02,40,A9,05,20,D2,FF,*31F
1064 DATA$C1F8,AE,3F,02,9A,4C,7B,E3,A0,*3D3
1065 DATA$C200,01,84,BA,84,B9,88,84,B7,*43F
1066 DATA$C208,84,90,84,93,A9,40,85,BB,*454
1067 DATA$C210,A9,02,85,BC,20,CF,FF,C9,*4A3
1068 DATA$C218,20,F0,F9,C9,0D,F0,38,C9,*4D0
1069 DATA$C220,22,D0,14,20,CF,FF,C9,22,*3DF
1070 DATA$C228,F0,10,C9,0D,F0,29,91,BB,*43B
1071 DATA$C230,E6,B7,C8,C0,10,D0,EC,4C,*53D
1072 DATA$C238,56,C3,20,CF,FF,C9,0D,F0,*4CD
1073 DATA$C240,16,C9,2C,D0,DC,20,F1,C2,*48A
1074 DATA$C248,29,0F,F0,E9,C9,03,F0,E5,*4B2
1075 DATA$C250,85,BA,20,CF,FF,C9,0D,60,*463
1076 DATA$C258,6C,30,03,6C,32,03,20,FF,*25F
1077 DATA$C260,C1,D0,D4,A9,05,20,D2,FF,*504
1078 DATA$C268,A9,00,20,58,C2,A5,90,29,*341
1079 DATA$C270,10,D0,C4,4C,B0,C0,20,FF,*47F
1080 DATA$C278,C1,C9,2C,D0,BA,20,E2,C2,*504
1081 DATA$C280,20,D2,C2,20,CF,FF,C9,2C,*497
1082 DATA$C288,D0,AD,20,E2,C2,A5,C1,85,*52C
1083 DATA$C290,AE,A5,C2,85,AF,20,D2,C2,*4FD
1084 DATA$C298,20,CF,FF,C9,0D,D0,98,A9,*4D5
1085 DATA$C2A0,05,20,D2,FF,20,5B,C2,4C,*37F
1086 DATA$C2A8,B0,C0,A5,C2,20,B1,C2,A5,*50F
1087 DATA$C2B0,C1,48,4A,4A,4A,4A,20,C9,*31A
1088 DATA$C2B8,C2,AA,68,29,0F,20,C9,C2,*3B7
1089 DATA$C2C0,48,8A,20,D2,FF,68,4C,D2,*449
1090 DATA$C2C8,FF,09,30,C9,3A,90,02,69,*336
1091 DATA$C2D0,06,60,A2,02,B5,C0,48,B5,*37C
1092 DATA$C2D8,C2,95,C0,68,95,C2,CA,D0,*570
1093 DATA$C2E0,F3,60,20,F1,C2,90,02,85,*43D
1094 DATA$C2E8,C2,20,F1,C2,90,02,85,C1,*46D
1095 DATA$C2F0,60,A9,00,85,2A,20,A7,C0,*33F
1096 DATA$C2F8,C9,20,D0,09,20,A7,C0,C9,*412
1097 DATA$C300,20,D0,0E,18,60,20,18,C3,*271
1098 DATA$C308,0A,0A,0A,0A,85,2A,20,A7,*19E
1099 DATA$C310,C0,20,18,C3,05,2A,38,60,*282
1100 DATA$C318,C9,3A,90,02,69,08,29,0F,*23E
```



```
1101 DATA$C320,60,A2,02,2C,A2,00,B4,C1,*347
1102 DATA$C328,D0,08,B4,C2,D0,02,E6,26,*42C
1103 DATA$C330,D6,C2,D6,C1,60,20,A7,C0,*516
1104 DATA$C338,C9,20,F0,F9,60,A9,00,8D,*468
1105 DATA$C340,00,01,20,35,C3,20,F8,C2,*2F3
1106 DATA$C348,20,E5,C2,90,09,60,20,A7,*387
1107 DATA$C350,C0,20,E2,C2,B0,DE,AE,3F,*4FF
1108 DATA$C358,02,9A,A9,05,20,D2,FF,A9,*3E4
1109 DATA$C360,3F,20,D2,FF,4C,B0,C0,20,*40C
1110 DATA$C368,B8,C5,CA,D0,FA,60,E6,C3,*61A
1111 DATA$C370,D0,02,E6,C4,60,A2,02,B5,*435
1112 DATA$C378,C0,48,B5,27,95,C0,68,95,*436
1113 DATA$C380,27,CA,D0,F3,60,A5,C3,A4,*520
1114 DATA$C388,C4,38,E9,02,B0,0E,88,90,*3BD
1115 DATA$C390,0B,A5,28,A4,29,4C,9C,C3,*350
1116 DATA$C398,A5,C3,A4,C4,38,E5,C1,85,*533
1117 DATA$C3A0,1E,98,E5,C2,A8,05,1E,60,*388
1118 DATA$C3A8,20,3D,C3,20,D2,C2,20,4E,*342
1119 DATA$C3B0,C3,20,75,C3,20,4E,C3,20,*36C
1120 DATA$C3B8,98,C3,20,D2,C2,90,15,A6,*45A
1121 DATA$C3C0,26,D0,64,20,91,C3,90,5F,*3BD
1122 DATA$C3C8,A1,C1,81,C3,20,6E,C3,20,*417
1123 DATA$C3D0,9C,C0,D0,EB,20,91,C3,18,*4A3
1124 DATA$C3D8,A5,1E,65,C3,85,C3,98,65,*430
1125 DATA$C3E0,C4,85,C4,20,75,C3,A6,26,*431
1126 DATA$C3E8,D0,3D,A1,C1,81,C3,20,91,*464
1127 DATA$C3F0,C3,B0,34,20,21,C3,20,24,*2EF
1128 DATA$C3F8,C3,4C,E6,C3,20,3D,C3,20,*3F8
1129 DATA$C400,D2,C2,20,4E,C3,20,D2,C2,*479
1130 DATA$C408,20,A7,C0,20,F1,C2,90,14,*3FE
1131 DATA$C410,85,1D,A6,26,D0,11,20,98,*307
1132 DATA$C418,C3,90,0C,A5,1D,81,C1,20,*383
1133 DATA$C420,9C,C0,D0,EE,4C,56,C3,4C,*4CB
1134 DATA$C428,B0,C0,20,3D,C3,20,D2,C2,*444
1135 DATA$C430,20,4E,C3,20,D2,C2,20,A7,*3AC
1136 DATA$C438,C0,A2,00,20,A7,C0,C9,27,*3D9
1137 DATA$C440,D0,14,20,A7,C0,9D,10,02,*31A
1138 DATA$C448,E8,20,CF,FF,C9,0D,F0,22,*4BE
1139 DATA$C450,E0,20,D0,F1,F0,1C,8E,00,*45B
1140 DATA$C458,01,20,F8,C2,90,C6,9D,10,*3DE
1141 DATA$C460,02,E8,20,CF,FF,C9,0D,F0,*49E
1142 DATA$C468,09,20,F1,C2,90,B6,E0,20,*422
1143 DATA$C470,D0,EC,86,1C,A9,05,20,D2,*3FE
1144 DATA$C478,FF,20,BB,C5,A2,00,A0,00,*3E1
1145 DATA$C480,B1,C1,DD,10,02,D0,0C,C8,*405
1146 DATA$C488,E8,E4,1C,D0,F3,20,AA,C2,*537
1147 DATA$C490,20,B8,C5,20,9C,C0,A6,26,*3E5
1148 DATA$C498,D0,8D,20,98,C3,B0,DD,4C,*4B1
1149 DATA$C4A0,B0,C0,20,3D,C3,85,20,A5,*3DA
1150 DATA$C4A8,C2,85,21,A2,00,86,28,A9,*361
1151 DATA$C4B0,93,20,D2,FF,A9,18,85,1D,*3E7
1152 DATA$C4B8,20,CE,C4,20,2E,C5,85,C1,*40B
1153 DATA$C4C0,84,C2,C6,1D,D0,F2,A9,91,*525
1154 DATA$C4C8,20,D2,FF,4C,B0,C0,A0,2C,*479
1155 DATA$C4D0,20,2B,C1,20,B8,C5,20,AA,*373
1156 DATA$C4D8,C2,20,B8,C5,A2,00,A1,C1,*463
1157 DATA$C4E0,20,3D,C5,48,20,83,C5,68,*33A
1158 DATA$C4E8,20,99,C5,A2,06,E0,03,D0,*3D9
1159 DATA$C4F0,12,A4,1F,F0,0E,A5,2A,C9,*36B
1160 DATA$C4F8,E8,B1,C1,B0,1C,20,26,C5,*431
1161 DATA$C500,88,D0,F2,06,2A,90,0E,BD,*3D5
1162 DATA$C508,91,C7,20,09,C6,BD,97,C7,*462
1163 DATA$C510,F0,03,20,03,C6,CA,D0,D5,*451
1164 DATA$C518,60,20,31,C5,AA,E8,D0,01,*3D9
1165 DATA$C520,C8,98,20,26,C5,8A,86,1C,*397
1166 DATA$C528,20,B1,C2,A6,1C,60,A5,1F,*379
1167 DATA$C530,38,A4,C2,AA,10,01,88,65,*346
1168 DATA$C538,C1,90,01,C8,60,A8,4A,90,*3FC
1169 DATA$C540,0B,4A,B0,17,C9,22,F0,13,*30A
1170 DATA$C548,29,07,09,80,4A,AA,BD,3D,*2A7
1171 DATA$C550,C7,B0,04,4A,4A,4A,4A,29,*2CC
1172 DATA$C558,0F,D0,04,A0,80,A9,00,AA,*356
1173 DATA$C560,BD,84,C7,85,2A,29,03,85,*368
1174 DATA$C568,1F,98,29,8F,AA,98,A0,03,*354
1175 DATA$C570,E0,8A,F0,0B,4A,90,08,4A,*391
1176 DATA$C578,4A,09,20,88,D0,FA,C8,88,*415
1177 DATA$C580,D0,F2,60,B1,C1,20,26,C5,*49F
1178 DATA$C588,A2,01,20,67,C3,C4,1F,C8,*398
1179 DATA$C590,90,F1,A2,03,C0,04,90,F2,*46C
1180 DATA$C598,60,A8,B9,9E,C7,85,28,B9,*48C
1181 DATA$C5A0,DE,C7,85,29,A9,00,A0,05,*3A1
1182 DATA$C5A8,06,29,26,28,2A,88,D0,F8,*2F7
1183 DATA$C5B0,69,3F,20,D2,FF,CA,D0,EC,*51F
1184 DATA$C5B8,A9,20,2C,A9,0D,4C,D2,FF,*3C8
1185 DATA$C5C0,20,3D,C3,20,D2,C2,20,4E,*342
1186 DATA$C5C8,C3,20,D2,C2,A2,00,86,28,*3C7
1187 DATA$C5D0,A9,05,20,D2,FF,20,BB,C5,*43F
1188 DATA$C5D8,20,D6,C4,20,2E,C5,85,C1,*413
1189 DATA$C5E0,84,C2,20,E1,FF,F0,05,20,*45B
1190 DATA$C5E8,98,C3,B0,E9,4C,B0,C0,20,*4D0
1191 DATA$C5F0,3D,C3,A9,03,85,1D,20,A7,*315
1192 DATA$C5F8,C0,20,0A,C1,D0,F8,A5,20,*438
1193 DATA$C600,85,C1,A5,21,85,C2,4C,AF,*44E
1194 DATA$C608,C4,C5,28,F0,03,20,D2,FF,*495
1195 DATA$C610,60,20,3D,C3,20,D2,C2,8E,*3C2
1196 DATA$C618,11,02,A2,03,20,35,C3,48,*218
1197 DATA$C620,CA,D0,F9,A2,03,68,38,E9,*4C1
1198 DATA$C628,3F,A0,05,4A,6E,11,02,6E,*21D
1199 DATA$C630,10,02,88,D0,F6,CA,D0,ED,*4E7
1200 DATA$C638,A2,02,20,CF,FF,C9,0D,F0,*458
1201 DATA$C640,1E,C9,20,F0,F5,20,34,C7,*407
1202 DATA$C648,B0,0F,20,05,C3,A4,C1,84,*390
1203 DATA$C650,C2,85,C1,A9,30,9D,10,02,*390
1204 DATA$C658,E8,9D,10,02,E8,D0,DB,86,*4B0
1205 DATA$C660,28,A2,00,86,26,F0,04,E6,*350
1206 DATA$C668,26,F0,75,A2,00,86,1D,A5,*375
1207 DATA$C670,26,20,3D,C5,A6,2A,86,29,*2C7
1208 DATA$C678,AA,BC,9E,C7,BD,DE,C7,20,*54D
1209 DATA$C680,1D,C7,D0,E3,A2,06,E0,03,*422
1210 DATA$C688,D0,19,A4,1F,F0,15,A5,2A,*380
1211 DATA$C690,C9,E8,A9,30,B0,21,20,23,*39E
1212 DATA$C698,C7,D0,CC,20,25,C7,D0,C7,*506
1213 DATA$C6A0,88,D0,EB,06,2A,90,0B,BC,*3CA
1214 DATA$C6A8,97,C7,BD,91,C7,20,1D,C7,*477
1215 DATA$C6B0,D0,B5,CA,D0,D1,F0,0A,20,*50A
1216 DATA$C6B8,1C,C7,D0,AB,20,1C,C7,D0,*431
1217 DATA$C6C0,A6,A5,28,C5,1D,D0,A0,20,*3E5
1218 DATA$C6C8,D2,C2,A4,1F,F0,28,A5,29,*43D
1219 DATA$C6D0,C9,9D,D0,1A,20,85,C3,90,*448
1220 DATA$C6D8,0A,98,D0,04,A5,1E,10,0A,*253
1221 DATA$C6E0,4C,56,C3,C8,D0,FA,A5,1E,*4BA
1222 DATA$C6E8,10,F6,A4,1F,D0,03,B9,C2,*417
1223 DATA$C6F0,00,91,C1,88,D0,F8,A5,26,*46D
1224 DATA$C6F8,91,C1,20,2E,C5,85,C1,84,*42F
1225 DATA$C700,C2,A9,05,20,D2,FF,A0,41,*442
1226 DATA$C708,20,2B,C1,20,B8,C5,20,AA,*373
1227 DATA$C710,C2,20,B8,C5,A9,05,20,D2,*3FF
1228 DATA$C718,FF,4C,14,C6,A8,20,23,C7,*3D7
```

```
1229 DATA$C720,D0,11,98,F0,0E,86,1C,A6,*3BF
1230 DATA$C728,1D,DD,10,02,08,E8,86,1D,*29F
1231 DATA$C730,A6,1C,28,60,C9,30,90,03,*2D6
1232 DATA$C738,C9,47,60,38,60,40,02,45,*28F
1233 DATA$C740,03,D0,08,40,09,30,22,45,*1BB
1234 DATA$C748,33,D0,08,40,09,40,02,45,*1DB
1235 DATA$C750,33,D0,08,40,09,40,02,45,*1DB
1236 DATA$C758,B3,D0,08,40,09,00,22,44,*23A
1237 DATA$C760,33,D0,8C,44,00,11,22,44,*24A
1238 DATA$C768,33,D0,8C,44,9A,10,22,44,*2E3
1239 DATA$C770,33,D0,08,40,09,10,22,44,*1CA
1240 DATA$C778,33,D0,08,40,09,62,13,78,*241
1241 DATA$C780,A9,13,78,A9,00,21,81,82,*301
1242 DATA$C788,00,00,59,4D,91,92,86,4A,*299
1243 DATA$C790,85,9D,2C,29,2C,23,28,24,*212
1244 DATA$C798,59,00,58,24,24,00,1C,8A,*19F
1245 DATA$C7A0,1C,23,5D,8B,1B,A1,9D,8A,*30A
1246 DATA$C7A8,1D,23,9D,8B,1D,A1,00,29,*24F
1247 DATA$C7B0,19,AE,69,A8,19,23,24,53,*28B
1248 DATA$C7B8,1B,23,24,53,19,A1,00,1A,*189
1249 DATA$C7C0,5B,5B,A5,69,24,24,AE,AE,*368
1250 DATA$C7C8,A8,AD,29,00,7C,00,15,9C,*2AB
1251 DATA$C7D0,6D,9C,A5,69,29,53,84,13,*32A
1252 DATA$C7D8,34,11,A5,69,23,A0,D8,62,*350
1253 DATA$C7E0,5A,48,26,62,94,88,54,44,*2DE
1254 DATA$C7E8,C8,54,68,44,E8,94,00,B4,*3F8
1255 DATA$C7F0,08,84,74,B4,28,6E,74,F4,*3B2
1256 DATA$C7F8,CC,4A,72,F2,A4,8A,00,AA,*452
1257 DATA$C800,A2,A2,74,74,74,72,44,68,*3BE
1258 DATA$C808,B2,32,B2,00,22,00,1A,1A,*1EC
1259 DATA$C810,26,26,72,72,88,C8,C4,CA,*40E
1260 DATA$C818,26,48,44,44,A2,C8,00,0D,*26D
1261 DATA$C820,20,20,20,50,43,20,20,53,*186
1262 DATA$C828,52,20,41,43,20,58,52,20,*1E0
1263 DATA$C830,59,52,20,53,50,00,00,00,*16E
1264 DATA$C838,00,00,93,12,05,C3,4F,4D,*209
1265 DATA$C840,50,55,54,45,52,D3,43,48,*2EE
1266 DATA$C848,41,55,2D,CD,4F,4E,49,54,*2CA
1267 DATA$C850,4F,52,20,20,20,20,20,20,*161
1268 DATA$C858,28,43,41,4C,4C,2D,45,4E,*204
1269 DATA$C860,54,52,59,29,20,0D,C2,45,*25C
1270 DATA$C868,46,45,48,4C,45,3A,20,41,*1FF
1271 DATA$C870,20,44,20,46,20,47,20,48,*193
1272 DATA$C878,20,4C,20,4D,20,52,20,53,*1BE
1273 DATA$C880,20,54,20,58,20,5F,20,5E,*1E9
1274 DATA$C888,0E,0D,00,00,A9,7F,A2,04,*1E9
1275 DATA$C890,A0,00,20,BA,FF,A9,00,20,*342
1276 DATA$C898,BD,FF,20,C0,FF,B0,58,A9,*54C
1277 DATA$C8A0,00,85,FD,A9,04,85,FE,A2,*454
1278 DATA$C8A8,7F,20,C9,FF,A2,19,A9,0D,*3D8
1279 DATA$C8B0,20,D2,FF,A9,1D,A0,05,EA,*446
1280 DATA$C8B8,20,D2,FF,88,D0,FA,20,D2,*535
1281 DATA$C8C0,FF,20,E1,FF,F0,31,A0,00,*4C0
1282 DATA$C8C8,B1,FD,85,FC,29,3F,06,FC,*499
1283 DATA$C8D0,24,FC,10,02,09,80,70,02,*22D
1284 DATA$C8D8,09,40,20,D2,FF,C8,C0,28,*3EA
1285 DATA$C8E0,D0,E6,98,18,65,FD,85,FD,*54A
1286 DATA$C8E8,90,02,E6,FE,CA,D0,BF,A9,*578
1287 DATA$C8F0,0D,20,D2,FF,20,CC,FF,A9,*492
1288 DATA$C8F8,7F,20,C3,FF,A9,9D,20,D2,*499
1289 DATA$C900,FF,A9,2C,20,D2,FF,4C,B0,*4C1
1290 DATA$C908,C0,00,*C0

READY.
```


VC-20 C-64

COMPUTER FÜR KINDER

Ein Buch für Kinder und ihre Lehrer – ein kindgemäßes Buch für die erste Begegnung mit Computern, ihren Eigenwilligkeiten, und ihren unerschöpflichen Möglichkeiten.
Ein Buch zu unserer Gegenwart und zur Zukunft unserer Kinder.
„Computer für Kinder" richtet sich an Kinder im Alter von 8 bis 13 Jahren, für deren Interesse an Computern keines der unzähligen Computer-Bücher geschrieben wurde.
„Computer für Kinder" ist ganz auf Kinder eingestellt und beschäftigt sich unterhaltsam und leicht verständlich mit folgenden Themen:

Wie arbeiten Computer
Wie funktioniert mein Computer
Wie programmiert man mit einfachen Flußdiagrammen
Wie kann ich BASIC leicht verstehen
Programme aufbauen mit Befehlen
Farbige Graphiken entwerfen
Erklärung von Computer-Begriffen

Sally Greenwood Larson war Kindergärtnerin, ehe sie selbst Computern begegnete und zwischen den Welten von Kindern und Computern zu vermitteln begann.
Computer für Kinder, A4 quer, Fadenheftung, über 100 Seiten, je Ausgabe DM 29,80

te-wi
te-wi Verlag GmbH
Theo-Prosel-Weg 1
8000 München 40

Weiterführende Literatur...

NEU! C-64 Computerhandbuch
Ein Handbuch für jeden Erfahrungsstand: von der ersten Begegnung bis zum professionellen Einsatz des COMMODORE 64 bzw. 1541. Das Werk ist sehr bildreich und bietet somit eine schnelle Übersicht – als echtes Nachschlagewerk werden Sie es stets in der Nähe Ihres Computers finden.
Raeto West, ca. 400 Seiten, Softcover,
DM 56,—, 4. Qu. 84

NEU! C-64 Akustik und Graphik
Ein planvoller Lehrgang – keine Beispielsammlung – in anschaulichem Stil – daher für jedes Alter. Dieses Werk eröffnet dem C-64-Benutzer die Welt der Graphiken und Klangbilder. Es enthält Programmbibliotheken und wird abgerundet durch zahlreiche Anhänge.
John Anderson, ca. 200 Seiten, Softcover,
DM 49,—, 4. Qu. 84

6502 – Programmieren in Assembler
Dieses Buch behandelt ausführlich die Assemblersprachen-Programmierung für den weitverbreiteten Mikroprozessor 6502. Er steckt auch in Ihrem C-64.
Lance Leventhal, 704 Seiten, Softcover,
DM 59,—

Der sensible C-64
Eine Softwaresammlung zu den technologischen Neuerscheinungen im C-64. Für Erstbenutzer wie für Experten – ein Buch der Softwarenutzung aller technologischen Eigenheiten des C-64.
Highmore/Page, Softcover, DM 29,80

CBM Computer Handbuch
Dieses unentbehrliche Nachschlagewerk bietet eine wahre Fundgrube – mit einer schrittweisen Einführung bis hin zur Darstellung aller professionellen Möglichkeiten dieses beliebten Computers.
Osborne/Danahue, 544 Seiten, Softcover,
DM 59,—

NEU! LOGO Computersprache für Kinder und Eltern
Dieses Buch beweist: **Jeder kann programmieren.** LOGO ist die Computersprache für Eltern und Kinder. Nicht umsonst wurde dieser Titel zum „Buch des Jahres 1983" in den USA. LOGO ist das Ergebnis der Erforschung menschlicher Intelligenz; entwickelt von einem Pädagogen und Mathematikprofessor. LOGO ist die erste Computersprache, die bewußt Strategien menschlichen Denkens dient.
Daniel Watt, ca. 400 Seiten, Softcover,
DM 59,—, 4. Qu. 84

NEU!
C-64/IEEE-488 Buch und Steckmodul
Mit diesem Steckmodul schaffen Sie sich Mehrfachnutzung durch nur ein Interface, das speziell den C-64 an die CBM-Großperipherie führt. Hiermit haben Sie zugleich ein Werkzeug, das z.B. sämtliche Elemente professioneller Meß- und Regelsysteme Ihren Bedürfnissen zugänglicher macht.
40 Seiten plus Modul, DM 239,–

CP/M und WordStar	**DM 29,80**
C-64 Programmsammlung	**4. Q. 84, DM 29,80**
VisiCalc (mit CBM Diskette)	**DM 79,––**
77 BASIC Programme	**DM 39,––**
Mikrocomputer-Grundwissen	**DM 36,––**

tm 4092

Allgemeines

War zu Beginn der Home- und Personalcomputerzeit die Kassette das gebräuchlichste Massenspeichermedium, so hat sich dies in der letzten Zeit grundlegend geändert. Heute hat sich die Diskette fast überall durchgesetzt.
Die Gründe hierfür sind leicht einzusehen, wenn man sich ein paar der Vorteile gegenüber der Kassette vor Augen hält. So z.B.:

- höhere Schreib- und Lesegeschwindigkeit gegenüber den normalen Bandaufzeichnungsverfahren
- direkte Zugriffsmöglichkeiten auf alle Spuren und Sektoren und damit direkte Manipulation einzelner Bytes
- kein serieller Ablauf beim Suchen und Laden eines Programmes, was sich durch schnelle Zugriffszeiten äußert
- schnelle Ausgabe des Inhaltsverzeichnisses
- schnelles Verändern/Erweitern/Überschreiben von Datenfiles, Programmen usw.

Weiter beigetragen zur schnellen Verbreitung haben freilich auch die hohen Verkaufsstückzahlen der Computer und deren peripheren Systemen. Dadurch wurden hohe Losgrößen in der Fertigung erreicht. Dies erlaubte z.B. auch der Firma Commodore rationell und kostengünstig zu fertigen, was sich dann durch günstige Preise auf dem Markt auswirkte. Hiervon profitierte dann auch der C64-Benutzer, für den Floppysysteme in preislich interessante Größenordnungen gelangten.
Trotz dieser großen Verbreitung aber ist das Wissen der Anwender über die Diskettenstation (1541), das Diskettenformat und die Diskette selbst noch sehr gering.

Dies betrifft weniger die normale Handhabung wie das Laden und Speichern von Programmen, sondern mehr die Möglichkeiten, die sich mit mehr Detailwissen den Anwendern bieten.
Daß dies so ist, liegt nicht zuletzt auch an den meist zu geringen Informationen und Erklärungen, welche die Handbücher enthalten. Das gleiche Problem ist uns ja auch bestens von den Computern selbst bekannt. Nicht umsonst gibt es heute an fast jeder „Ecke" weiterführende „Computerliteratur" für alle möglichen Systeme zu kaufen.
Dieser Artikel soll demjenigen, der sich etwas näher mit seinen Disketten beschäftigen will, weiterhelfen. Wobei aber keinesfalls eine vollständige Dokumentation aller Einzelheiten erfolgen kann, denn damit könnte man ein ganzes Buch füllen.

Die Diskette selbst

Verwirrend für den Interessierten können die unterschiedlichen Namensgebungen dieser Speicherscheiben sein. Umgangssprachlich werden sie als „Standard-Diskette", „Mini-Diskette", „Floppy-Disk", „Standard flexible Disk" und wahrscheinlich noch mit einigen weiteren Wortschöpfungen bezeichnet.
Bei unserer zu betrachtenden Diskette handelt es sich um die 5,25-Zoll-Ausführung, welche mit dem Commodore-Format „2A" beschrieben wird.
Das „Herz", eine runde Kunststoffscheibe mit aufgebrachter, magnetisierbarer Eisenoxid-Schicht (angeblich sind auch welche mit Chromdioxidschicht erhältlich), befindet sich in einer mit Vlies ausgekleideten Schutzhülle. Das Vlies dient vor allem auch dazu, um Abriebverschmutzungen vom Schreib-/Lesekopf der Diskettenstation fernzuhalten.
In dieser Hülle verbleibt die eigentliche Speicherscheibe auch während der Benutzung, trotz der hohen Umdrehungsgeschwindigkeit von ca. 300 Umdrehungen pro Minute.
Damit der Diskettenstation der Zugriff zu den Daten möglich ist, sind auf der Ober- und Unterseite der Hülle zwei ovale Ausschnitte für den Schreib-/Lesekopf vorhanden.
Eine Bearbeitung erfolgt bei unserer bereits genannten Station aber nur einseitig, wobei wir bereits beim nächsten Punkt angelangt sind. Es gibt sogenannte „single sided"- und „double sided"-Ausführungen. Für uns relevant ist die „single sided – single density"-Ausführung, wobei „single density" darauf hinweist, daß mit „Einfacher Schreibdichte", im Gegensatz zur „Doppelten Schreibdichte", aufgezeichnet wird.
Daß eine „single sided – single Density"-Diskette gegebenfalls auch doppelseitig, mit doppelter Schreibdichte, funktionieren kann, soll hier nicht unerwähnt bleiben, aber garantiert wird dies vom Hersteller natürlich nicht. Der Grund für diese „Trotzdemfunktionsfähigkeit" ist oft nur ein nicht bestandener Qualitätstest im gesamten Fertigungslos, welcher aber nicht alle Exemplare betreffen muß. Umgekehrt klappt es dann natürlich schon, nur sind diese Disketten etwas teuerer.
Über das Aussehen und den Aufbau gibt Bild 1 Auskunft.

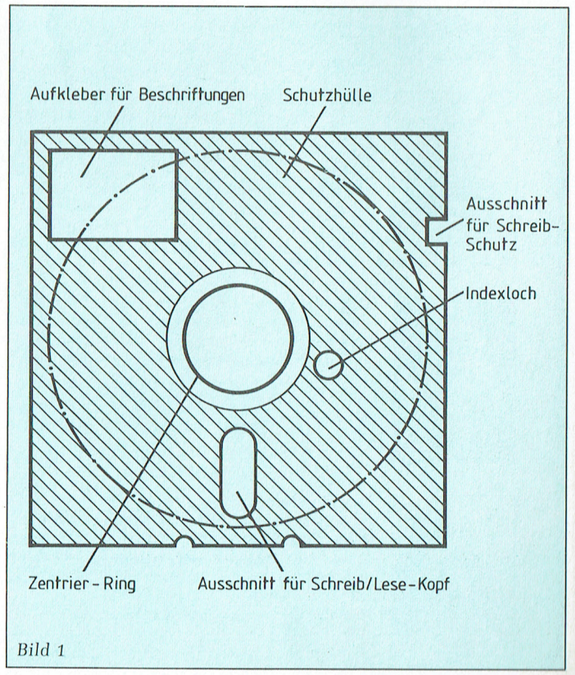


Bild 1

Sinn der Verstärkungsringe

Als praktisch haben sich Ausführungen mit „Zentrierring" bewährt, da diese mechanisch stabiler sind und das Mittelloch nicht so schnell „ausleiert".
Beim Verfasser hat sich gezeigt, daß neue Disketten, deren Innenlochtoleranzen etwas hoch waren (bedingt vielleicht auch noch durch Abnutzungserscheinungen des Antriebs-Anpreßkegelstumpfes in der Diskettenstation) enorme Schwierigkeiten bereiteten. Nach der Nachrüstung mit Zentrierringen wurden diese Disketten in der Mitte höher und die Probleme waren behoben. Sie ließen sich dann einwandfrei beschreiben und lesen.

Formatierung erforderlich

Im „Rohzustand" befinden sich die Magnetpartikel der Speicherschicht in einem völlig ungeordneten Zustand. Das heißt, keine Floppystation kann damit etwas „Lesbares" anfangen. Um nun mit der Diskette arbeiten zu können, muß dieser erst das Format „aufgedruckt", d.h. sie muß „formatiert" werden. Diese Formatierung erfolgt „softsektoriert", was nichts anderes bedeutet, als daß ein magnetisches Muster aufgeschrieben wird. Im Gegensatz hierzu wird bei der „Hardsektorierung" im wahrsten Sinne des Wortes „gelöchert"!

Der Formatierungsvorgang bewirkt, daß die Diskette ihren Namen und ihre ID (Identifikationskennzeichnung) bekommt und außerdem in Spuren (englisch: Tracks) und Sektoren (engl. Sector), auch Blöcke genannt, aufgeteilt wird. Siehe hierzu Bild 2.

Dabei wird die ID in jedem „Header" und auch in Spur 18 abgelegt. Als Header bezeichnet man den Vorspann, welcher sich vor den eigentlichen Speicherdaten eines jeden Blockes befindet.

Unterschiedliche Blockanzahlen

Wie in Bild 2 zu erkennen ist, sind die äußeren „Ringe" (Spuren) größer als die inneren. Dies bedingt aber dann, daß bei optimaler Platzausnutzung auf den inneren Spuren weniger Sektoren untergebracht werden können als auf den äußeren. So ist es auch bei unserem Format. Dies liegt durch das „DOS" (Disketten-Operating-System) fest.
In Zahlen ausgedrückt, sieht es folgendermaßen aus:

Wozu mehr Kenntnisse?

Selbst wenn man nicht vorhat, großartige Leistungen in Verbindung mit der Diskette zu vollbringen, ist es dennoch unerläßlich mehr zu wissen, wenn aus irgendwelchen Gründen Probleme auftreten. Sei es nun, daß man aus Versehen ein Programm gelöscht hat, daß Datenfiles nicht mehr komplett lesbar sind, daß ein Archivierprogramm für die File-Namen nicht richtig arbeitet (weil zwei Disketten beispielsweise die gleiche ID haben), daß man den Diskettennamen ohne Programmverlust umbenennen will oder man sich vielleicht selbst mal helfen muß, weil der „Spezialist", den man sonst immer gerne

● Spur 1...17	je 21 Blöcke (0...20) =	357 Blöcke
● Spur 18...24	je 19 Blöcke (0...18) =	133 Blöcke
● Spur 25...30	je 18 Blöcke (0...17) =	108 Blöcke
● Spur 31...35	je 17 Blöcke (0...16) =	85 Blöcke
Insgesamt: 35 Spuren		683 Blöcke

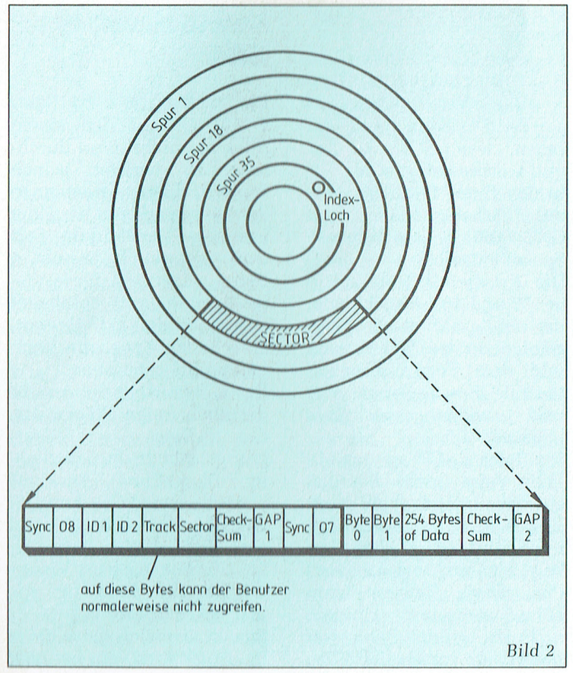


Bild 2

Die Gesamtanzahl der Blökke kann jedoch vom Anwender normalerweise nicht voll benützt werden, denn die Spur 18 (also etwa in der Mitte der Kunststoffscheibe) hat eine besondere Funktion. Diese Spur trägt die Informationen über die Blockbelegung, den Diskettennamen, die Namen der „Files" (also die Programm- und Datei-Namen), die ID usw. Sie kann aber noch mehr beinhalten, hierzu jedoch später. Wichtig zu wissen ist, daß uns – im Regelfall – die Spur 18 nicht frei zur Verfügung steht.
Da die Spur 18 aus 19 Blökken besteht und jeder Block aus 256 Datenbytes, stehen dem Benutzer damit 664★256 Bytes (also rund 166 KBytes) als Speicherplatz zur Verfügung. Wie bereits vermerkt, gilt dies nur für Normalanwendungen.

bemüht, gerade nicht greifbar ist. Wenn man ein stundenlanges „Neueintippen" vermeiden will, weil man vergessen hat, ein File richtig zu schließen, weil man ein File gegen „Löschen" schützen will usw., usw.
Es gibt viele Gründe, sich etwas eingehender mit der Thematik „Diskette" zu beschäftigen.

Das DOS braucht Headerhilfe

Zurück zu unseren Sektoren, welche durch die Formatierung erzeugt wurden. Weiter oben war bereits angeführt, daß ein Block aus 256 Bytes besteht. Dies sind die Datenbytes, zu welchen wir Zugriff haben. Aber dies reicht dem Betriebssystem der 1541 noch nicht aus, um sinnvolle Tätigkeiten ausführen zu können. Deswegen befindet sich vor jedem Block noch der Header, in

welchem sich – für das DOS wichtige – Daten befinden.
Der Header beinhaltet mehrere Informationen, wie z.B. Synchronisationszeichen, die ID (2 Bytes), Spur- und Sektor-Nummer usw. Diese erst ermöglichen es dem Disk-Controller, einen bestimmten Sektor zu finden, Fehler zu erkennen usw. Üblicherweise hat der Anwender keinen direkten Zugriff auf diesen Vorspann und er ist nur durch „Memory-Execute"-Befehle manipulierbar.
Bei professionellen Programmen wird dies oftmals ausgenutzt, um einen Kopierschutz zu realisieren, indem absichtlich die Header „verändert" werden.
Hinter dem, dem Header folgenden, eigentlichen Datenblock ist zusätzlich noch eine Prüfsumme aufgezeichnet.
Wie schon erwähnt, hat die Spur 18 eine Sonderfunktion, deshalb wollen wir uns diese Spur nun etwas näher betrachten.

Die Spur 18

Läßt man sich das Inhaltsverzeichnis einer neuformatierten Diskette ausgeben, so sieht dies in etwa wie Bild 3a aus.

```
0 "FRANZIS - VERLAG" CS 2A
664 BLOCKS FREE.

BILD 3A
```

In der ersten, der reversen Zeile, steht (in Anführungszeichen) der Disketten-Name, gefolgt von der ID und dem Aufzeichnungsformat. Die 0 am Anfang weist lediglich darauf hin, daß es sich um das Inhaltsverzeichnis des Laufwerkes 0 handelt. Da noch keine weiteren Programme bzw. Files aufgezeichnet wurden, ist darunter die Gesamtzahl der insgesamt verfügbaren Blöcke zu lesen, nämlich 664.
Sieht man sich mit einem „Disk-Monitor", wie z.B. dem Programm „Display T&S" den Block 0 der Spur 18 an, so erfolgt (abhängig von diesem Programm bzw. Drucker) eine Darstellung, ähnlich dem Bild 3b.

```
TRACK 18  SECTOR 0

00 :12 01 41 00 15 FF FF 1F 15 FF FF 1F 15 FF FF 1F :  A  ▯▯  ▯▯  ▯▯
10 :15 FF FF 1F 15 FF FF 1F 15 FF FF 1F 15 FF FF 1F : ▯▯  ▯▯  ▯▯  ▯▯
20 :15 FF FF 1F 15 FF FF 1F 15 FF FF 1F 15 FF FF 1F : ▯▯  ▯▯  ▯▯  ▯▯
30 :15 FF FF 1F 15 FF FF 1F 15 FF FF 1F 15 FF FF 1F : ▯▯  ▯▯  ▯▯  ▯▯
40 :15 FF FF 1F 15 FF FF 1F 11 FC FF 07 13 FF FF 07 : ▯▯  ▯▯  ▯▯  ▯▯
50 :13 FF FF 07 13 FF FF 07 13 FF FF 07 13 FF FF 07 : ▯▯  ▯▯  ▯▯  ▯▯
60 :13 FF FF 07 12 FF FF 03 12 FF FF 03 12 FF FF 03 : ▯▯  ▯▯  ▯▯  ▯▯
70 :12 FF FF 03 12 FF FF 03 12 FF FF 03 11 FF FF 01 : ▯▯  ▯▯  ▯▯  ▯▯
80 :11 FF FF 01 11 FF FF 01 11 FF FF 01 11 FF FF 01 : ▯▯  ▯▯  ▯▯  ▯▯
90 :46 52 41 4E 5A 49 53 20 2D 20 56 45 52 4C 41 47 :FRANZIS - VERLAG
A0 :A0 A0 43 53 A0 32 41 A0 A0 A0 A0 00 00 00 00 00 :  CS 2A
B0 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
C0 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
D0 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
E0 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
F0 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :

BILD 3 B
```

Ein derartiger Monitor wird mit der „Systemdiskette" mitgeliefert und ist unter dem Namen „Display T&S" abgelegt. Wenn Sie allerdings mehr sehen und tun wollen, dann sollten Sie beispielsweise das von der Firma Interface Age erhältliche Programm „EXDOS&DISK-DOCTOR" einsetzen.

Sector 0

Die ersten beiden Bytes (Byte 0 und Byte 1) dieses Blokkes stellen die „Linkadressen", also die Verweiszeiger, auf den nächsten logischen Block dar. In diesem Beispiel zeigen sie auf den Beginn des Inhaltsverzeichnisses (englisch: Directory). Da im Bild 3b die Darstellung hexadezimal (sedezimal) erfolgt, muß schnell umgerechnet werden. Schon geschehen? ... Richtig, Spur 18, Sektor 1, dort beginnt das Inhaltsverzeichnis.
Denn hexadezimal 12 = 1 ★ 16 + 2 = 18.
Das 3. Byte (also Byte 2!) hat den Inhalt 41. Dies ist das ASCII-Zeichen für „A" und läßt das Diskettenaufzeichnungsformat erkennen.
Das Byte 3 mit dem Inhalt „00" wird derzeit nicht benützt, es ist für evtl. zukünftige DOS-Versionen (Disk-Operating-System) vorgesehen.
Die Bytes 4...143 stellen die BAM (Block-Avaibility-Map = Block-Verfügbarkeits-Liste) dar. Das heißt, hier ist zu sehen, welche Blöcke frei und welche belegt sind.
In den Bytes 144...161 steht der Disketten-Name, mit Chr$ (160) (= „geshiftetes" Space) aufgefüllt.
Die Disketten-ID steht in Byte 162 und 163. Danach wieder ein „geshiftetes" Space, gefolgt von der DOS-Version und dem Format-Kennzeichen. DOS = 2, Format = A und wieder geshiftete „Spaces".
Die Bytes ab 171 werden üblicherweise nicht benützt, aber ab und zu steht dort: „Bytes free". In diesem Bereich kann nun einiger „Sinn bzw. Unsinn" abgelegt werden. Diese Zeichen sind beim normalen Arbeiten nicht zu sehen, denn wer macht sich schon die Mühe, jede seiner Disketten auf „versteckte" Zeichen hin zu untersuchen. Bei professionellen Programmen sind dort gelegentlich Copyright-vermerke und ähnliches zu finden. Aber auch Programme können dort „nachsehen", ob bestimmte Kennzeichen abgelegt sind, um ggf. die Arbeit zu verweigern, falls dies nicht der Fall ist.

Block-Availability-Map (BAM)

Die Struktur der BAM (Bytes 4-143) ist auf den ersten Blick hin nicht sofort durchschaubar, bereitet jedoch keine Probleme, wenn man das System erkannt hat. Zum besseren Verständnis faßt man gedanklich jeweils 4 Bytes zusammen. Diese Vierer-Kombination beinhaltet dann, jeweils für die entsprechende Spur, die kompletten Informationen.
Das 1. Byte stellt die Anzahl der noch freien Blöcke dar, das 2. Byte dieser Vierergruppe gibt in hexadezimaler Darstellung Auskunft über die freien Blöcke 0...7, danach folgen 8...15 und zum Schluß die Sektoren 16...20. Die nächste Vierergruppe gilt dann für die nächste Spur etc.
Schaut man sich diese Bytes in „Bitstruktur" an, hat man bereits das Ergebnis vorliegen. Ein bißchen Gedankenarbeit wollen wir dem Leser nun auch noch lassen, aber schwierig ist es bestimmt nicht!

Directory

Für das Inhaltsverzeichnis (Directory) sind ab Block 1 alle weiteren Blöcke reserviert bzw. vorgesehen. Die ersten beiden Bytes jedes Sektors teilen immer mit, welcher Block der nächstfolgende logische Sektor ist.
Der letzte logische Block enthält als Byte Null die hexadezimale Zahl „00". Daraus erkennt das DOS, daß dies der letzte Block des Inhaltsverzeichnisses ist. Würde dort beispielsweise „12" und im nachfolgenden Byte „02" stehen, dann würde dies bedeuten, daß das Inhaltsverzeichnis in Spur 18, Sektor 2, weitergeht. Wie sieht es nun aus, wenn Files auf der Diskette gespeichert sind?
Wieder ist es der Eintrag im Byte 2, der uns Auskunft gibt. Ist dieses Byte = 0, dann ist dies ein Hinweis darauf, daß dies wieder der letzte Block ist. Das nächste Byte ist das Kennzeichen, daß es sich um ein gültiges Programm handelt (siehe hierzu auch die Aufstellung über die File-Typen) und die beiden Folge-Bytes teilen mit, in welcher Spur und in welchem Sektor das Programm beginnt. Danach folgt der File-Name wieder ergänzt durch „A0" (=chr$(160)), also geshiftetes Space, und in den Bytes 30 und 31 steht, wieviele Blöcke das Programm belegt. In unserem Beispiel fünf Blöcke.
Die Kennzeichen der Filetypen:

Filetyp	nein	ja
Schutz gegen „Scratch"		
DELeted	80	C0
SEQuential	81	C1
PRoGramm	82	C2
USeR	83	C3
RELative	84	C4

Wie zu sehen ist, kann durch entsprechende Bitmanipulation ein File vor dem Löschen geschützt werden. Dies wird im normalen Inhaltsverzeichnis durch das Zeichen „<" hinter der File-Art gekennzeichnet. Durch direkte Zugriffe auf diese Bytes kann dann beispielsweise auch ein „geschütztes, nicht geschlossenes" File oder ähnliche sinnlose Kombinationen erzeugt werden. Wie sich derartiges aber in letzter Konsequenz dann auswirkt, wurde von uns noch nicht im Detail untersucht. Trotzdem, das Ganze ist doch sehr einfach, oder?
Damit möchten wir die Spur 18 abschließen, der Interessierte kann ja mittels der Direktzugriffsbefehle bzw. mittels Diskmonitor noch mehr

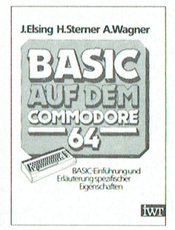
```
0 "FRANZIS - VERLAG" CS 2A
5     "ID-WECHSLER"        PRG
659 BLOCKS FREE.

BILD 4
```

Die Betrachtung des Bildes 4 (Inhalt der nun mit einem Programm beschriebenen Diskette) zeigt, daß ein Programm mit dem Namen „ID-Wechsler" 5 Blöcke belegt, wodurch dann nur noch 659 Blöcke frei sind.
Wie Bild 4b zeigt, sind auf Spur 18, Sektor 0 kleine Änderungen gegenüber Bild 3b eingetreten, welche die Anzahl der freien Blöcke betreffen. In Spur 18, Sektor 1 können Sie nun den Namen des gespeicherten Programmes und noch weitere Informationen erhalten (siehe Bild 4c).
Doch nun wieder der Reihe nach. In den ersten beiden Bytes ist kenntlich gemacht,

Neues aus Vaterstetten:

Mit dem iwt-Programm auf die Zukunft programmiert!

Der C 64 bietet vielseitige grafische Möglichkeiten. Dieses Buch gibt Informationen wie man Grafikfunktionen anwendet. Ausgehend von Grafiken mit den ›festen‹ Grafik-Zeichen wird systematisch zu den anspruchsvolleren Möglichkeiten, illustriert durch typische Beispiele, geführt.
138 Seiten. 1 Folie. Spiralh.
DM 38,–/Fr. 38.–



Systematische Einführung in BASIC. Außer vielen kleineren Programmen zur Illustrierung der BASIC-Anweisungen gibt es eine umfangreiche Programmsammlung zu verschiedenen Themen. Die Fähigkeiten des C64 werden mit vielen Programmbeispielen erläutert.
356 Seiten. Spiralh.
DM 56,–/Fr. 56.–

Dieses Buch behandelt den Einsatz des deutschsprachigen IWT LOGO auf dem C 64. Anhand vieler ausführlicher Beispiele aus Mathematik, Geometrie, Grammatik und Musik wird der Einsatz von IWT LOGO im Unterricht erläutert.
Ca. 200 Seiten. Spiralh.
DM 44,–/Fr. 44.–

Dieses Buch zeigt, wie sich komplizierte Operationen verständlich beschreiben lassen. Es wird demonstriert, wie einfach sich dreidimensionale Probleme lösen lassen. Die Beispiele reichen von der Geraden über das Dreikörperproblem bis hin zum dreidimensionalen Planetensystem.
208 Seiten. Kart.
DM 44,–/Fr. 44.–

Die Programmierung des Video Interface Chips 6567 ist Hauptthema des Buches. Basic - Grafikprogramme werden von Maschinenprogrammen zum Punkt-/Linienzeichnen unterstützt, was die Schnelligkeit vielfach erhöht, teilweise Basic-Programme direkt in Maschinensprache parallel dargestellt.
152 Seiten. Spiralh.
DM 38,–/Fr. 38.–

Grafikprogramme werden ›gehirngerecht‹ aufbereitet. Neue Art des Formats – man bekommt ein ›Bild‹ des Befehls, Demo-Programme unterstützen das Gedächtnis, Bildschirm-Hardcopies als schnelles Nachschlagewerk, farbige Übersichtskarten zur Programmier-Erleichterung.
208 Seiten. Spiralh.
DM 44,–/Fr. 44.–

Ich bin neugierig auf Ihr Gesamtprogramm! Senden Sie mir umgehend

☐ Ihren neuesten Computer- und Elektronik-Literaturkatalog.
☐ Erbitte Unterlagen über Ihr umfangreiches Software-Programm.
☐ Ich interessiere mich für Ihre ROBOTIK-Idee.
☐ Ich möchte mit D.A.T.A.BOOKS Zeit und Geld sparen.

Name/Vorname
Firma
Abt.
Straße/Hausnr.
PLZ/Ort

IWT Verlag, Vaterstetten
Der Fachverlag für Information, Wissenschaft, Technologie
Dahlienstraße 4, 8011 Baldham, Tel. (08106) 31017, Tx 5213989 iwt
Auslieferung Schweiz: Thali AG, Buchhandlung und Verlag, CH-6285 Hitzkirch, Tel. (041) 85 28 28
Auslieferung Österreich: Oberösterreichischer Landesverlag Linz, Fachbuchabteilung, Landstr. 41, A-4010 Linz, Tel. (07 32) 27 81 21/296/245, Tx 02/1014

```
TRACK 18  SECTOR 0

00 :12 01 41 00 15 FF FF 1F 15 FF FF 1F 15 FF FF 1F :  A  ‹‹  ‹‹  ‹‹
10 :15 FF FF 1F 15 FF FF 1F 15 FF FF 1F 15 FF FF 1F : ‹‹  ‹‹  ‹‹  ‹‹
20 :15 FF FF 1F 15 FF FF 1F 15 FF FF 1F 15 FF FF 1F : ‹‹  ‹‹  ‹‹  ‹‹
30 :15 FF FF 1F 15 FF FF 1F 15 FF FF 1F 15 FF FF 1F : ‹‹  ‹‹  ‹‹  ‹‹
40 :15 FF FF 1F 10 FE FA 0B 11 FC FF 07 13 FF FF 07 : ‹‹  _|  "‹  ‹‹
50 :13 FF FF 07 13 FF FF 07 13 FF FF 07 13 FF FF 07 : ‹‹  ‹‹  ‹‹  ‹‹
60 :13 FF FF 07 12 FF FF 03 12 FF FF 03 12 FF FF 03 : ‹‹  ‹‹  ‹‹  ‹‹
70 :12 FF FF 03 12 FF FF 03 12 FF FF 03 11 FF FF 01 : ‹‹  ‹‹  ‹‹  ‹‹
80 :11 FF FF 01 11 FF FF 01 11 FF FF 01 11 FF FF 01 : ‹‹  ‹‹  ‹‹  ‹‹
90 :46 52 41 4E 5A 49 53 20 2D 20 56 45 52 4C 41 47 :FRANZIS - VERLAG
A0 :A0 A0 43 53 A0 32 41 A0 A0 A0 A0 00 00 00 00 00 :  CS 2A
B0 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
C0 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
D0 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
E0 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
F0 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :


BILD 4 B
```

```
TRACK 18  SECTOR 1

00 :00 FF 02 11 00 49 44 2D 57 45 43 48 53 4C 45 52 : ‹■  ID-WECHSLER
10 :A0 A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 05 00 :
20 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
30 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
40 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
50 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
60 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
70 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
80 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
90 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
A0 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
B0 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
C0 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
D0 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
E0 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
F0 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :


BILD 4 C
```

in der Spur 18 „herumstöbern“ und ggf. nach Belieben verändern.

Programm-Ablage

Wir wissen nun also bereits, wie man herausfinden kann, wo ein bestimmtes Programm beginnt und dort machen wir mal weiter. Zuerst aber zum Programm selbst, das Listing zeigt Bild 5. Es handelt sich um ein Programm zur Änderung der Disketten-ID. Diese wird aber – durch das Programm – nur in Spur 18, Sektor Null, nicht aber in den Headern geändert!

Laut unserem Inhaltsverzeichnis in Spur 18, Sektor 1, befindet sich der erste Teil des Programmes in Spur 17, Sektor 0. Byte Nr. 0–1 dieses Sektors beinhalten wieder die nächste Blockadresse, damit sich der Disk-Controller „durchhangeln“ kann.

Danach folgt die Startadresse (im RAM des Rechners; im Beispiel $0401, da auf einem CBM 3032 geschrieben!), gefolgt von der Linkadresse zur nächsten Zeile, dann die Zeilen-Nummer (in Low-Byte-/High-Byte-Darstellung) und dann..., ja dann folgt das Programm selbst. Über die Linker ist ja zu verfolgen, über welche Blöcke das Ganze verteilt ist. Bis letztendlich zu den abschließenden 3 Bytes mit dem Inhalt „00“, welche das Programmende darstellen. Wer sich schon einmal damit befaßt hat, wie Programme im Rechner „stehen“, wird hier sofort erkennen können, daß bis auf ein paar Bytes alles identisch ist.

Arbeiten mit Diskette

```
100 PRINT"■ID-WECHSLER "
110 :
120 REM KOMMANDOKANAL OEFFNEN UND INITIALISIEREN
130 OPEN15,8,15,"I0"
140 :
150 REM PUFFER OEFFNEN UND KANAL 3 ZUORDNEN
160 OPEN1,8,3,"#"
170 :
180 REM LESEZUGRIFF DRIVE 0   = U1
190 REM               KANAL 3
200 REM               LAUFWERK 0
210 REM               SPUR 18
220 REM               SEKTOR 0
230 PRINT#15,"U1:3"0",18,0"
240 :
250 REM BLOCK-POINTER SETZEN
260 REM               KANAL 3
270 REM               BYTE 162
280 PRINT#15,"B-P:3,162"
290 :
300 REM 2 BYTES HOLEN
310 FORI=1TO2
320 GET#1,X$
330  ID$=ID$+X$
340 NEXT
350 :
360 PRINT"■ALTE DISK-ID :  ■ "ID$
370 :
380 REM AUSGABE ALTE/EINGABE NEUE ID
390 INPUT"■NEUE DISK-ID : ";NI$
400 :
410 REM LAENGE DEFINIEREN
420 IFLEN(NI$)<>2THENNI$=LEFT$(NI$+"  ",2)
430 :
440 REM BLOCK-POINTER WIEDER SETZEN
450 PRINT#15,"B-P:3,162"
460 :
470 REM NEUE ID IN PUFFER SCHREIBEN
480 PRINT#1,NI$;
490 :
500 REM SCHREIBAUSFUEHRUNG
510 PRINT#15,"U2:3"0",18,0
520 PRINT#15,"I0"
530 CLOSE1:CLOSE15
540 :
550 REM NEUE ID HOLEN
560 OPEN15,8,15:PRINT#15,"I0":OPEN1,8,3,"#":PRINT#15,"U1:3"0",18,0"
570 PRINT#15,"B-P:3,162"
580  ID$=""
590 GET#1,A$,B$
600 PRINT"■GEAENDERTE ID   :■ "A$B$
610 CLOSE1:CLOSE15
READY.
```

Bild 5

Die Marke IWT SOFTWARE der Profis

iwt Logo®

›Wem die (Deutsch-)Stunde schlägt‹, dachte der Computer und lernte eilends um.

IWT Logo ...

- die leicht zu erlernende Computer-Sprache für jede Altersstufe
- die Computer-Sprache mit modernen Strukturen, Prozeduren, lokalen Variablen, Listen usw.
- die Computer-Sprache mit der »Igel-Grafik«, die von Anfang an zu sichtbaren Erfolgen führt
- die Computer-Sprache, die sofort auf den Kern der Sache – das Programmieren und Erproben von Programmen – führt

... natürlich in deutsch
(Befehle, Fehler- und Systemmeldungen in deutscher Sprache)

IWT Logo	
– für Apple][//e, 2c Best.-Nr. 400 01 101	DM 395,–*
– deutsche Erweiterung zum C64 Logo, Version C64 105 Best.-Nr. 400 22 101	DM 98,–*

*incl. MwSt./unverbindl. Preisempfehlung

IWT Software Service – für Information, Wissenschaft, Technologie
Technologie-Zentrum: Höhestraße 66, Postfach 1325, 5093 Burscheid 1,
Tel. (02174) 62815, Tx 5213989 iwt · Vertrieb/Beratung: Dahlienstr. 4,
Postfach 100243, 8011 Baldham, Tel. (08106) 31017, Tx 5213989 iwt

iwt

CS

```
TRACK 17  SECTOR 0

00 :11 0A 01 04 16 04 64 00 99 22 93 49 44 2D 57 45 :       — ■"▒▼■ID-WE
10 :43 48 53 4C 45 52 20 22 00 1C 04 6E 00 3A 00 4B :CHSLER "   / : K
20 :04 78 00 8F 20 4B 4F 4D 4D 41 4E 44 4F 4B 41 4E : ♣ ■ KOMMANDOKAN
30 :41 4C 20 4F 45 46 46 4E 45 4E 20 55 4E 44 20 49 :AL OEFFNEN UND I
40 :4E 49 54 49 41 4C 49 53 49 45 52 45 4E 00 5D 04 :NITIALISIEREN ]
50 :82 00 9F 31 35 2C 38 2C 31 35 2C 22 49 30 22 00 :▌ ◣15,8,15,"I0"
60 :63 04 8C 00 3A 00 8D 04 96 00 8F 20 50 55 46 46 :— ■ : ■ ■ ■ PUFF
70 :45 52 20 4F 45 46 46 4E 45 4E 20 55 4E 44 20 4B :ER OEFFNEN UND K
80 :41 4E 41 4C 20 33 20 5A 55 4F 52 44 4E 45 4E 00 :ANAL 3 ZUORDNEN
90 :9C 04 A0 00 9F 31 2C 38 2C 33 2C 22 23 22 00 A2 :▒   ◣1,8,3,"#" ▬
A0 :04 AA 00 3A 00 C2 04 B4 00 8F 20 4C 45 53 45 5A :  ▌: ▌▌ ■ LESEZ
B0 :55 47 52 49 46 46 20 44 52 49 56 45 20 30 20 20 :UGRIFF DRIVE 0
C0 :3D 20 55 31 00 DC 04 BE 00 8F 20 20 20 20 20 20 := U1 ▒ " ■
D0 :20 20 20 20 20 20 20 4B 41 4E 41 4C 20 33 00 F9 :       KANAL 3 ▬
E0 :04 C8 00 8F 20 20 20 20 20 20 20 20 20 20 20 20 :  ▌■
F0 :20 4C 41 55 46 57 45 52 4B 20 30 00 13 05 D2 00 : LAUFWERK 0   _
```

```
TRACK 17  SECTOR 10

00 :11 14 8F 20 20 20 20 20 20 20 20 20 20 20 20 20 :  ■
10 :53 50 55 52 20 31 38 00 2E 05 DC 00 8F 20 20 20 :SPUR 18 . ▒ ■
20 :20 20 20 20 20 20 20 20 20 20 53 45 4B 54 4F 52 :          SEKTOR
30 :20 30 00 45 05 E6 00 98 31 35 2C 22 55 31 3A 33 : 0 E ▒ ◪15,"U1:3
40 :22 30 22 2C 31 38 2C 30 22 00 4B 05 F0 00 3A 00 :"0",18,0" K ┌ :
50 :66 05 FA 00 8F 20 42 4C 4F 43 4B 2D 50 4F 49 4E :_ ┘ ■ BLOCK-POIN
60 :54 45 52 20 53 45 54 5A 45 4E 00 80 05 04 01 8F :TER SETZEN ▀   ■
70 :20 20 20 20 20 20 20 20 20 20 20 20 20 4B 41 4E :                KAN
80 :41 4C 20 33 00 9B 05 0E 01 8F 20 20 20 20 20 20 :AL 3 ▪▪   ■
90 :20 20 20 20 20 20 20 42 59 54 45 20 31 36 32 00 :       BYTE 162
A0 :AF 05 18 01 98 31 35 2C 22 42 2D 50 3A 33 2C 31 :_   ◪15,"B-P:3,1
B0 :36 32 22 00 B5 05 22 01 3A 00 C9 05 2C 01 8F 20 :62" ▌ " : ╮ , ■
C0 :32 20 42 59 54 45 53 20 48 4F 4C 45 4E 00 D4 05 :2 BYTES HOLEN |
D0 :36 01 81 49 B2 31 A4 32 00 DF 05 40 01 A1 23 31 :6 ◪I┬1_2 ◥ @ ▌#1
E0 :2C 58 24 00 EE 05 4A 01 49 44 24 B2 49 44 24 AA :,X$ ╮ J ID$┬ID$ ▌
F0 :58 24 00 F4 05 54 01 82 00 FA 05 5E 01 3A 00 18 :X$ ▌ T ▌▌ ┘ ↑ :
```

```
TRACK 17  SECTOR 20

00 :11 08 06 68 01 99 22 11 41 4C 54 45 20 44 49 53 :    ▌ ■" ALTE DIS
10 :4B 2D 49 44 20 3A 20 20 12 20 22 49 44 24 00 1E :K-ID :    "ID$
20 :06 72 01 3A 00 41 06 7C 01 8F 20 41 55 53 47 41 : _ : A ▒ ■ AUSGA
30 :42 45 20 41 4C 54 45 2F 45 49 4E 47 41 42 45 20 :BE ALTE/EINGABE
40 :4E 45 55 45 20 49 44 00 5D 06 86 01 85 22 11 4E :NEUE ID ] ▀ ▀" N
50 :45 55 45 20 44 49 53 4B 2D 49 44 20 3A 20 22 3B :EUE DISK-ID : ";
60 :4E 49 24 00 63 06 90 01 3A 00 7B 06 9A 01 8F 20 :NI$ — ■ : + ◘ ■
70 :4C 41 45 4E 47 45 20 44 45 46 49 4E 49 45 52 45 :LAENGE DEFINIERE
80 :4E 00 9B 06 A4 01 8B C3 28 4E 49 24 29 B3 B1 32 :N ▪▪ _ ▒▬(NI$)┤±2
90 :A7 4E 49 24 B2 C8 28 4E 49 24 AA 22 20 22 2C 32 : NI$┬|(NI$ ▌" ",2
A0 :29 00 A1 06 AE 01 3A 00 C3 06 B8 01 8F 20 42 4C :) ▌ ╮ : — ▀ ■ BL
B0 :4F 43 4B 2D 50 4F 49 4E 54 45 52 20 57 49 45 44 :OCK-POINTER WIED
C0 :45 52 20 53 45 54 5A 45 4E 00 D7 06 C2 01 98 31 :ER SETZEN o | ◪1
D0 :35 2C 22 42 2D 50 3A 33 2C 31 36 32 22 00 DD 06 :5,"B-P:3,162" |
E0 :CC 01 3A 00 FF 06 D6 01 8F 20 4E 45 55 45 20 49 :└ : ƒ X ■ NEUE I
F0 :44 20 49 4E 20 50 55 46 46 45 52 20 53 43 48 52 :D IN PUFFER SCHR
```



```
TRACK 17  SECTOR 8

00 :11 12 45 49 42 45 4E 00 0B 07 E0 01 98 31 2C 4E :  EIBEN     ◪1,N
10 :49 24 3B 00 11 07 EA 01 3A 00 2A 07 F4 01 8F 20 :I$;    ▌: * ▌ ■
20 :53 43 48 52 45 49 42 41 55 53 46 55 45 48 52 55 :SCHREIBAUSFUEHRU
30 :4E 47 00 40 07 FE 01 98 31 35 2C 22 55 32 3A 33 :NG @   ◪15,"U2:3
40 :22 30 22 2C 31 38 2C 30 00 4D 07 08 02 98 31 35 :"0",18,0 M   ◪15
50 :2C 22 49 30 22 00 58 07 12 02 A0 31 3A A0 31 35 :,"I0" X    1: 15
60 :00 5E 07 1C 02 3A 00 72 07 26 02 8F 20 4E 45 55 : ↑   : _ & ■ NEU
70 :45 20 49 44 20 48 4F 4C 45 4E 00 A6 07 30 02 9F :E ID HOLEN ▒ 0 ◣
80 :31 35 2C 38 2C 31 35 3A 98 31 35 2C 22 49 30 22 :15,8,15:◪15,"I0"
90 :3A 9F 31 2C 38 2C 33 2C 22 23 22 3A 98 31 35 2C ::◣1,8,3,"#":◪15,
A0 :22 55 31 3A 33 22 30 22 2C 31 38 2C 30 22 00 BA :"U1:3"0",18,0" ┘
B0 :07 3A 02 98 31 35 2C 22 42 2D 50 3A 33 2C 31 36 : : ◪15,"B-P:3,16
C0 :32 22 00 C5 07 44 02 49 44 24 B2 22 22 00 D3 07 :2" — D ID$┬"" ♥
D0 :4E 02 A1 28 31 2C 41 24 2C 42 24 00 F2 07 58 02 :N ▌#1,A$,B$ ┬ X
E0 :99 22 11 47 45 41 45 4E 44 45 52 54 45 20 49 44 :▌" GEAENDERTE ID
F0 :20 20 3A 12 20 22 41 24 42 24 00 FD 07 62 02 A0 :  :  "A$B$ ┘ |
```

```
TRACK 17  SECTOR 18

00 :00 0A 31 3A A0 31 35 00 00 00 00 00 00 00 00 00 :  1: 15
10 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
20 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
30 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
40 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
50 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
60 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
70 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
80 :4E 00 9B 06 A4 01 8B C3 28 4E 49 24 29 B3 B1 32 :N ▪▪ _ ▒▬(NI$)┤±2
90 :A7 4E 49 24 B2 C8 28 4E 49 24 AA 22 20 22 2C 32 : NI$┬|(NI$ ▌" ",2
A0 :29 00 A1 06 AE 01 3A 00 C3 06 B8 01 8F 20 42 4C :) ▌ ╮ : — ▀ ■ BL
B0 :4F 43 4B 2D 50 4F 49 4E 54 45 52 20 57 49 45 44 :OCK-POINTER WIED
C0 :45 52 20 53 45 54 5A 45 4E 00 D7 06 C2 01 98 31 :ER SETZEN o | ◪1
D0 :35 2C 22 42 2D 50 3A 33 2C 31 36 32 22 00 DD 06 :5,"B-P:3,162" |
E0 :CC 01 3A 00 FF 06 D6 01 8F 20 4E 45 55 45 20 49 :└ : ƒ X ■ NEUE I
F0 :44 20 49 4E 20 50 55 46 46 45 52 20 53 43 48 52 :D IN PUFFER SCHR
```

Wie nun andere Filetypen strukturiert sind, kann ja jeder für sich selbst herausfinden, indem er seine Disketten etwas näher untersucht. Es wird bestimmt ein interessantes Wandern über Spuren und Blöcke und trägt gewiß dazu bei, mehr über seine Massenspeicher zu erfahren.

Mit dem Wissen über das Diskettenformat, dem Wissen, wie der Disk-Controller arbeitet etc. kann man dann auf der Diskette „schalten und walten", wie man will. Man kann beispielsweise Start- und Endadressen von Programmen verändern, Texte innerhalb von Programmen ändern, bereits gelöschte Programme wieder zurückgewinnen, vorausgesetzt, es war vorher noch kein weiterer schreibender Zugriff. Kann die entsprechenden Blöcke als wieder belegt kennzeichnen, kann erreichen, daß Files anscheinend einige tausend Blöcke belegen usw. Erforderlich ist lediglich, daß man die „direkten Zugriffsmöglichkeiten" beherrscht.

Ein kleines Beispiel ist das Programm: „Id-Wechsler", welches als Basis für weitere Möglichkeiten dienen kann. Schreiben Sie dieses Programm doch so um, daß Sie auch den Disketten-Namen verändern können, ohne die Diskette neu formatieren zu müssen, wodurch ja sonst alle Fileeinträge verloren gehen würden. Was genau bei einer Umformatierung ohne Angabe einer ID passiert, was wirklich geschieht, wenn ein File gelöscht wird, wenn ein File durch ein neues mit gleichem Namen ersetzt wird usw., ist bestimmt nicht weniger interessant zu erfahren, als zum Beispiel auch das Arbeiten mit Direktzugriffs-Dateien, das Abarbeiten von „Block-Execut"-Befehlen, das direkte Ansteuern der Schrittmotore in der Floppy usw.

Wußten Sie schon, daß man auch außerhalb der Spur 35 arbeiten kann?

Daß man auch zwischen den Spuren arbeiten kann?

Daß man ein ganz anderes Aufzeichnungsformat, auch mit der 1541 „fahren" kann?

Daß man durch „Rückweisen" eines Linkers auf der Diskette ein „ewiges" Inhaltsverzeichnis bekommt, welches sich immer wieder wiederholt?

Daß Programm-Namen auf der Diskette oft ganz anders aussehen, als im Directory-Listing?

Dieses und noch viel mehr können Sie sehen und lernen, wenn Sie sich mit Ihrer Floppy etwas beschäftigen.

Schlußbemerkungen Diskette

Nur einen wirklich kleinen Teil konnten wir bezüglich der Disketten bzw. der Diskettenstation aufzeigen, wie bereits bei einigen Artikeln dieses Heftes mitgeteilt: ganze Bücher könnte man füllen mit alldem, was eben nicht im Handbuch steht.
Derjenige aber, der sich wirklich interessiert, wie sein Computer, seine Floppystation, sein Drucker usw. funktionieren, muß selbst arbeiten. Denn Erfolge kann man sich nicht erlesen, man muß sie erarbeiten.
Abschließend noch einige Anmerkungen zu bisher nicht Angesprochenem beim Arbeiten und Umgang mit Disketten. Wir haben zwar schon darauf hingewiesen, daß dann, wenn das erste Byte eines Blockes eine Null beinhaltet, dies der letzte Block ist, aber über das Folgebyte haben wir uns noch nicht weiter geäußert. Dieses gibt dann nämlich an, wieviele Bytes des letzten Blokkes wirklich benutzt wurden. Wenn Sie sich mehrere Ihrer Disketten mal etwas genauer ansehen, werden Sie feststellen, daß dann, wenn auch der letzte Block nur ein paar Zeichen beinhaltet, trotzdem dahinter anstelle von Nullen auch andere Zeichen enthalten sein können. Woran das liegt, können Sie sich durch verschiedene kleine Experimente leicht selbst erklären.
Sie werden feststellen können, daß im Inhaltsverzeichnis auch der Name von bereits gelöschten Programmen noch vorhanden ist. Lediglich der Filetyp wurde auf Null gesetzt.
In diesem Zusammenhang auch gleich einmal bemerkt: In jedem Block der Spur 18, von Sektor 1 an aufwärts, können maximal 8 Fileeinträge stehen, aus diesem Grunde können auf einer Diskette nur insgesamt 144 Fileeinträge vorhanden sein! Eingangs haben wir erwähnt, daß in Sektor 0, Spur 18 oberhalb des Bytes 170 also, von 171 bis 256, Zeichen unterschiedlicher Art stehen können, die aber für das DOS völlig unwichtig sind. Dort beispielsweise kann man Versions-Nummern u. a. unterbringen.
Wenn man Files durch Anwendung des Klammeraffen (@) überschreibt, geben zwei Bytes Track und Sector des neuen Eintrages an. Versuchen Sie diese mal herauszufinden. Sehen Sie sich ruhig auch einmal an, wie seq. Files abgelegt werden, oder aber, und das ist dann schon etwas schwieriger, wie es bei relativen Files aussieht. Bei diesen wird ein sogenannter „Side-Sector" angelegt, der aus bis zu sechs Blöcken bestehen kann. Diese Blöcke stellen dann die Zeiger auf die einzelnen „Datenblöcke" dar.
Diejenigen, die auch die Möglichkeit haben, von den Diskettenstationen
CBM 3040 oder 4040 formatierte Disketten näher unter die Lupe nehmen zu können, sollten dies ruhig einmal tun, denn Sie werden dann feststellen, daß trotz des gleichen Formates (= 2A) Disketten von diesem System im Verhältnis zur 1541 anders aufbereitet werden.
Sie sehen, ein großes Feld zum Spielen und Lernen tut sich auf, man muß nur wollen. Es müssen also nicht immer fremde Datenbanken sein, in denen man sein Unwesen treibt, versuchen Sie es doch einmal mit Ihren eigenen.

Allgemeine Hinweise:

Prinzipiell sollte jede Diskette eine eigene, von anderen unterschiedliche ID haben, damit das DOS einen Diskettenwechsel immer erkennen kann. Seien Sie vorsichtig mit Magneten in Diskette-Nähe, auch Lautsprecher in Fernsehgeräten enthalten Magnete.
Bewahren Sie Ihre Disketten immer in der dafür vorgesehenen Tasche auf und entnehmen Sie sie nur bei Bedarf.

Und als letztes noch ein kleiner Typ:

Falls Sie Leseprobleme mit fremden Disketten haben, überprüfen Sie ruhig mal die Geschwindigkeit Ihrer 1541. Dazu brauchen Sie diese nur umzudrehen. Netz-Leuchtstoffröhre – wegen der 50 Hz – einschalten, dann können Sie sogar im zusammengebauten Zustand überprüfen, was Ihnen die Stroboskopscheibe mitteilt.

Arbeiten mit den Block-Kommandos

Liest man in den Handbüchern und auch in den anderen zum C64 und zur Floppy 1541 erschienen Büchern nach, so erscheint einem der Umgang mit den Block-Kommandos der 1541 sehr einfach, und auch sehr schnell in seine eigenen Programme einbaubar. Speziell haben wir uns das Kommando für die Belegtkennzeichnung eines Blockes herausgegriffen, das es bei Direktzugriffsdateien ermöglicht, bestimmte Blöcke vor dem schreibenden Zugriff zu schützen und auch Schutz beim Abspeichern von Programmen, USR- oder auch SEQ-Files bieten sollte. Bis dahin ist auch alles in Ordnung.

Der Autor, der schon im Umgang mit der CBM 4040 einige Erfahrungen hat, war sehr erstaunt, als er in verschiedenen Büchern nun eine Möglichkeit sah, wie man derartige Befehle handhabt. Es sei nochmals ausdrücklich erwähnt, daß die gleiche Art der Handhabung nicht nur in einem Buch zu lesen war!

Also das Programm abgeschrieben und versucht zu starten.

Probleme

Es schien bestens zu funktionieren. Abgesehen davon, daß immer wieder als Antwort kam, daß Track 0 Sektor 0 belegt wurden. Solange man immer wieder die Spur wechselte, lief die Floppy auch an und es schien so, daß der Block wirklich als belegt gekennzeichnet wurde.
Blieb man aber innerhalb einer Spur, dann wurden die gewünschten Blöcke zwar anscheinend im DOS-Speicher, aber nicht auf der Diskette belegt.
Dies bedeutet aber, daß erst ein weiterer Zugriff auf die Floppy erfolgen muß, der dann erst wirklich die Blökke belegt. Nun wollten wir es genau wissen!

Also eine völlig neu formatierte in das Laufwerk eingeschoben.

1. Run, Eingabe von Track 1, Sector 1

Meldung des Programmes: „Track 0 Sektor 0 wurde belegt.

So weit, so gut. Nun wurde die Floppy aus- und wieder eingeschaltet. Das Ausgeben der dann eingelesenen Directory zeigte, daß noch 643 Blöcke frei waren. Wie aber das, wenn doch eigentlich nur ein Block belegt werden sollte? Das Vertrauen in die Computerei sank nun doch beträchtlich. Trotzdem wurde der Computer nun zum Rechnen eingesetzt.

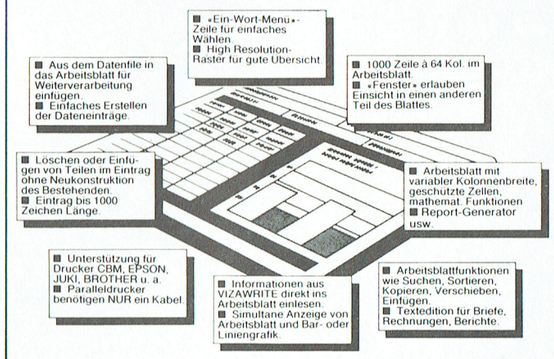
Einfache Aufgabe:
664 – 1 = 663
Nachberechnung:
664 – 643 = 21
Aha!!! Hier lag der Fehler!
Statt eines Blockes wurden 21 als belegt gekennzeichnet!
Nächster Versuch:
2. Run, Eingabe: Track 27, Sector 1
Meldung des Programmes: wie vorher!
Floppy ausgeschaltet, Neueinschaltung, Directory einlesen, ausgeben, noch 646 Blocks frei! Unser Verdacht bestätigte sich. Nachberechnung: 664 – 646 = 18
Trotzdem, zur Sicherheit noch einen Versuch.
3. Run, Eingabe: Track 20, Sector 5
Meldung des Programmes: wie vorher!
Floppy aus und wieder ein usw. Verdacht nun vollends bestätigt!
Noch 645 Blöcke frei!
Nachberechnung:
664 – 645 = 19
Nun wußten wir ganz sicher, was eigentlich passiert war: Anstelle eines einzigen Blockes wurde die komplette Spur belegt!

Beweis:
Track 01 hat 21 Spuren
Track 27 hat 18 Spuren
Track 20 hat 19 Spuren

Nun sollte aber noch ein letzter Versuch folgen.
4. Run, Eingabe: Track 31, Sector 1
Meldung des Prorammes wie vor.
Ergebnis: 17 belegte Blöcke!

Anstelle eines einzigen Blockes wurde tatsächlich jeweils die komplette Spur belegt. Dies erklärt auch, weshalb dann, wenn man in einer Spur mehrere Blöcke belegen wollte, immer die zermürbende Antwort „65 no block 02 00" kam, wenn wir nach T/S 1,1 auch noch 1,2 belegen wollten. Wir hatten nämlich zwischenzeitlich auf ein etwas einfacheres Programm umgestellt, welches sich eine Seite vorher in dem schon angeführten Buch befand.
Ja, und dann haben wir eben unser eigenes Programm geschrieben, welches eigentlich auch hätte funktionieren müssen.
Es funktionierte zwar weitaus besser, d. h., wenn ein Block belegt werden sollte, dann wurde auch nur einer belegt. Aber wenn einer freigegeben werden sollte, dann waren plötzlich mehr Blöcke belegt als vorher!
Die Frage, warum wir all das geschrieben haben, ohne Ihnen derzeit einen Weg aufzeigen zu können, wie diese Fehler vermieden werden, dürfen Sie ruhig stellen. Wir wollten darauf hinweisen, daß im DOS „Bugs" sind, die manchem Programmierer Stunden, ja Tage kosten können, bis er dahinterkommt, weshalb sich seine Direktzugriffs-Dateien plötzlich verändern.
Wir werden die Sache noch weiterverfolgen und sobald wir den Grund für dieses eigenartige Verhalten und auch die entsprechenden Gegenmaßnahmen haben, werden wir dies in einem unserer regulären Computer-Schau-Hefte veröffentlichen.

VIZASTAR 64
Informationsverarbeitung
■ Kalkulation ■ Datenbank ■ Grafik



VIZASTAR integriert drei elektronische Hilfen in Ihrem Geschäfts- oder Heimbüro. Das leistungsfähige Informationsverarbeitungssystem beinhaltet ein elektronisches Arbeitsblatt für Kalkulationen, ein Datenbank-Eingabesystem und Bildschirmgrafik. VIZASTAR ist sehr benutzerfreundlich ausgebaut und ganz einfach für automatischen Ablauf zu programmieren.

VIZASTAR ist in seiner Konzeption das einzige Programm dieser Art für den Commodore 64. Selbstverständlich ist der deutsche Zeichensatz auf dem Bildschirm darstellbar. VIZASTAR ist kompatibel zum Textverarbeitungsprogramm VIZAWRITE 64.
Verlangen Sie den 8seitigen VIZASTAR-Prospekt.
(Vertrieb auch für Vizawrite)

Vertrieb Deutschland:
INTERFACE AGE
Verlag GmbH
Josefsburgstraße 6
8000 München 80
Tel.: 089/43 40 89

Vertrieb Schweiz:
MICROTRON
Computerprodukte
Bahnhofstrasse 2
CH-2542 Pieterlen

VIZA SOFTWARE

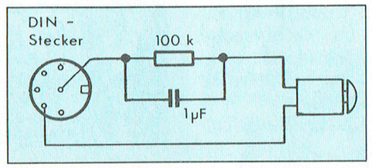
1. Resetschalter

Der C64 hat leider keinen eingebauten Resetknopf und durch die Betätigung von RUN/STOP und RESTORE kommt man auch nicht immer aus einem „Harakiri-Lauf" des Computers. Aber die Möglichkeit, über einen extern anzubringenden Schalter den Rechner evtl. wieder zur Vernunft zu bringen, besteht durchaus.

Hierzu ist es nicht einmal erforderlich, einen Eingriff in den Computer selbst vorzunehmen. Doch bevor wir sagen, wie man's macht, wollen wir erst einmal erklären, wie es funktioniert. Keine Angst, wir wollen nun nicht den kompletten C64 beschreiben, aber zum Verständnis kann dieser kleine Exkurs bestimmt ein wenig beitragen.
Um nach dem Einschalten im Computer genau definierte Anfangszustände zu erzeugen, wird, hervorgerufen durch das IC U20, einem Baustein mit zwei Timer-Stufen (Bezeichnung NE 556), über den Inverter U8 (7406) der Reset-Eingang des Prozessors kurzzeitig nach 0 V „gezogen". Dies geschieht nicht gleichzeitig mit dem Einschaltvorgang, sondern etwas verzögert, damit alle Versorgungsspannungen usw. erst korrekt anstehen, bevor der Prozessor durch den Signalübergang von HIGH nach LOW mit seiner Arbeit beginnt. Beim Start der Prozessortätigkeit holt sich dieser aus den Adressen $fffc und $fffd den „Resetvektor", also die Adresse, bei der er mit seiner Arbeit beginnt.
Falls man sich den Schaltplan des C64 etwas näher ansieht, kann man feststellen, daß die Resetleitung auch direkt zur seriellen Bus-Buchse und auch zum User-Port geführt wird. Ja, selbst am Cartridge-Expansion-Stecker ist über das Bauteil FB4 die Leitung zugänglich.
Das heißt, daß wir durch einen Taster das gleiche durchführen können, wie dies durch die Timerkombination geschieht. Allerdings mit dem Unterschied, daß nicht die gesamte Resetroutine, sondern nur ein Teil davon abgearbeitet wird, und deshalb keine Löschung der RAM-Speicherzellen oberhalb von 2052 geschieht. Die Wirkung ist die gleiche wie bei der Eingabe des Befehls Sys64738.

Reset ohne Programmverlust

Das bedeutet, daß ein im Speicher stehendes Programm nicht gelöscht wird, sondern lediglich die drei ersten Bytes ab 2049 auf Null gesetzt werden. Das aber wiederum heißt, daß man z.B. einen vorher in den Rechner geladenen Maschinensprache-Monitor auch nach einem „Taster-Reset" wieder starten kann, ja sogar ein Basic-Programm, welches durch einen falschen Sys-Befehl den Absturz bewirkte, wieder restaurieren und auch (nun zur Sicherheit) wieder abspeichern kann.



Eine direkte Abarbeitung des geretteten Basicprogrammes ist nicht anzuraten, da die Pointer nicht mehr stimmen! Durch Einsatz eines „OLD-Programmes" oder Korrektur von Hand können diese aber auch wieder richtiggestellt werden. Ein Maschinenprogramm benötigt keine Restaurierung der Pointer, sondern lediglich die Wiederherstellung der ersten Bytes, falls es beim normalen Basicstart liegt. Programme, welche unterhalb von 2048 dezimal liegen, werden leider gelöscht. Voraussetzung für das richtige Funktionieren ist, daß der C64 nicht so „Harakiri" lief, daß auch dieser Reset nicht mehr hilft. Denn dann gibt es wirklich nur einen „Netzneustart". Einen Versuch ist es allemal wert, wenn man nicht wieder alle Arbeit von vorne beginnen will.

Kurzschluß = Reset

Nun zur Realisierung: Wie Sie gesehen haben, braucht die Resetleitung nur kurzzeitig nach 0 V, also gegen Masse, gezogen werden. Im einfachsten Falle geschieht dies durch einen Kurzschluß. Da dieser aber nicht dauernd anliegen soll, genügt es, die Kontakte eines Tasters (Schließer) an die entsprechenden Leitungen anzuschließen, um dann beim Resetwunsch diesen kurzzeitig anzutippen. Nach Informationen aus den USA soll es aber durch diese direkte Technik schon zu Schäden an den C64 gekommen sein. Wir können uns zwar nicht erklären weshalb, aber vielleicht wurden irgendwelche Schaltungsänderungen durchgeführt, die uns nicht bekannt sind. Deswegen also bauen wir zwei zusätzliche Bauelemente ein: nämlich einen Widerstand und einen Kondensator. Der Widerstand sollte einen Wert in der Größenordnung von ca. 100 kΩ und der Kondensator von ca. 1 µF haben. Falls ein Elko (Elektrolyt-Kondensator) eingesetzt werden soll, ist auf seine Polarität (richtige Anschlußweise) zu achten. Beide Bauteile werden parallel und dann in Serie zum Taster geschaltet. Siehe Skizze! Die komfortabelste Art dieses kleinen Zusatzes finden wir als Einbau in einen kleinen sechspoligen DIN-Stecker, der sowohl den Resetknopf als auch die beiden Bauteile aufnehmen kann. Diese Resetkombination kann nämlich einfach auf den seriellen Bus sowohl am C64 als auch an der Floppy 1541 oder auch am Drucker eingesetzt werden und belegt dort keine evtl. benötigte Anschlußbuchse. Die beiden Leitungen am seriellen Bus liegen an den Kontakten Pin 6 (Reset) und Pin 2 (GND). Wer will, kann einen Resetknopf auch am Userport oder am Steckkontakt für die Module anbringen. Auch in den Rechner selbst ist der Einbau möglich. Die nötigen Informationen hierzu haben wir ja eben gegeben.
Bedenken Sie aber, daß durch einen direkten Eingriff in Ihren C64 die Garantie erlöschen kann.

Die Anschlußpunkte in Kürze:

	Reset	GND
Userport-Pin	3	1/12/A od. N
Expansion-Stecker	C	A/Z/1 oder 22
Serielle Bus-Buchse	6	2

Anmerkung: Es ist softwaremäßig möglich, auch einen Reset mittels Taster zu verhindern, hierzu ist uns aber bis heute leider noch kein Gegenmittel eingefallen.

2. Normaler Kassettenrecorder als Datenspeicher für den C64

Darüber, ob man mit der Kassettenspeicherung von Programmen und Datenfiles gut beraten ist, ist schon sehr viel geschrieben worden. Es gibt Anhänger für, aber auch gegen den Kassetten-(Datasetten-)Einsatz.

Tatsache aber ist, daß einige Programme nur in dieser Version angeboten werden. Will man also eines dieser Programme erwerben, kommt man nicht umhin, diese mittels Recorder einzulesen.

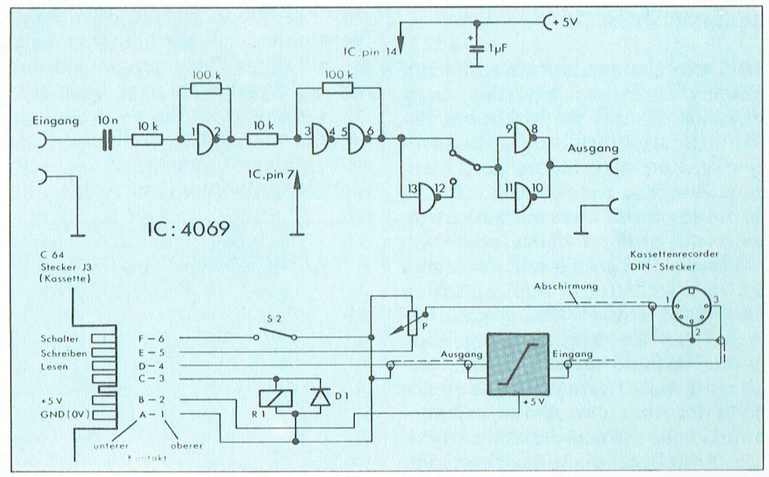
Die Kassette hat ihre Berechtigung

Der Autor dieses Artikels hat – als Sicherheitskopien – alle seine Diskettenprogramme und Datenfiles (auch Direktzugriffsdateien) in Kassettenformat gesichert, um bei Beschädigung des Originals wieder eine komplette Diskette „aufbauen" zu können. Die Aufzeichnungen und auch das Einlesen geschehen dabei mittels schneller Kassettenprogramme (also Verfahren, wie sie bei Turbo-Tape, Supertape usw. verwandt werden. Mittels dieser Programme ist die Kassette teilweise sogar schneller als die Floppy 1541!
Beim C64 (dies gilt auch für die anderen Commodore-Computer) kann aufgrund der Schnittstellenauslegung leider kein normaler Kassettenrecorder eingesetzt werden. So ist jeder, der mittels Kassetten arbeiten will, im Regelfall genötigt, sich eine Datasette (vielleicht auch leihweise) zu beschaffen.

Ein normaler Recorder genügt!

Dies ist aber gar nicht erforderlich. In unserer Schwesterzeitschrift „mc" erschien nämlich im Januar-Heft 1983 (Seite 36) eine kleine Schaltung von Alfred Schön, die es ermöglicht, einen normalen Kassettenrecorder als Daten- und Programmspeichersystem einzusetzen. In der Septemberausgabe (Seite 54) erschien dazu sogar noch ein Layout für eine Platine und weitere Informationen.

Dieses kleine Interface arbeitet beim Verfasser – von Anbeginn an – zur vollsten Zufriedenheit. Allerdings wurde es dahingehend erweitert, daß der Computer auch die Steuerung des Recorders selbst übernimmt. Dies geschieht derart, daß der Computer (C64, Pet 2001, CBM 3032...8032) über sein Kassettenrecorder-Anschluß-Port, Pin C oder 3, ein kleines Reed-Relais schaltet, welches seinerseits dann über den Fernbedienanschluß (Remote) den Recorder ein- und ausschaltet. Dies ist deswegen erforderlich, weil sonst bei mehrteiligen Programmen der Kassettenrecorder weiterlaufen würde, während der Rechner arbeitet, und dann beim nächsten Ladevorgang seinen weiteren Teil schon „überlesen" hätte.
Da wir nicht voraussetzen, daß jeder unserer Leser diese Hefte vorliegen hat, bringen wir hier in Kurzform nochmals die kleine Schaltung mit den entsprechenden Ergänzungen:
Wer sich weiter informieren will, sollte versuchen, die beiden weiter oben genannten Artikel zu Rate zu ziehen. Denn wir wollen uns, seitens des Verlages, ja nicht wiederholen.



3. Mithörverstärker für Load und Save mit Kassette

Bestimmt hat jeder Benutzer von Kassettenrecordern für Computer schon mal im Ungewissen darüber geschwebt, was denn eigentlich der Computer während der gewünschten Lese-/Schreiboperation tut. Bei „Schreibarbeiten" ist dies weniger von Interesse, aber beim Lesen... ja, da wäre es manchmal interessant, zu wissen, was denn augenblicklich „läuft". Ob Leseprobleme vorliegen, ob gerade ein Header gelesen wurde, ob das Programm bereits mit seinem zweiten Durchlauf gelesen wird usw.

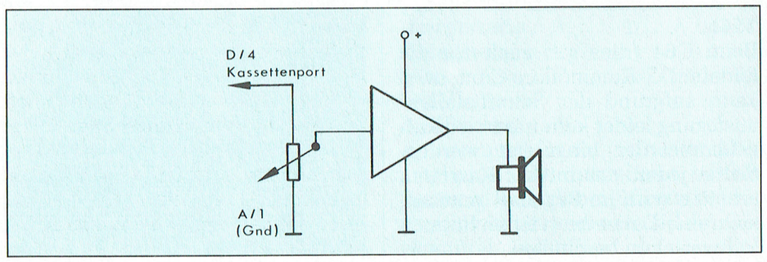
Ein Mithören bei diesen Vorgängen kann zwar durch Poke 54296,15 erreicht werden, jedoch sind die dann zu hörenden Töne doch recht unklar und dadurch nicht so einwandfrei identifizierbar.

Geringer Aufwand

Mit sehr geringem Aufwand kann diesem Umstand abgeholfen werden. Ein kleiner Verstärker mit Potentiometer (Einsteller für die Lautstärke) und angeschlossenem Lautsprecher oder Kopfhörer macht das Zuhören, wenn zwar auch nicht gerade zu einem musikalischen, so doch unter Umständen zu einem gut auswertbaren Vergnügen. Man hört nach einiger Übung ganz genau, was im Augenblick gelesen wird, bemerkt Leseprobleme, bedingt durch „Dropouts" oder Bandmaterialfehler usw. Auch die Justierung des Recorderkombikopfes kann erfolgen, wenn der Verstärker an den Schreib-Lese-Kopf direkt angeschlossen wird (hochohmiger Verstärkereingang und hohe Verstärkung erforderlich!). Es ist sogar möglich, Programme von einem C64 zu einem anderen (über Draht, Funk) zu überspielen. Hierbei sind aber die gesetzlichen Bestimmungen und Vorschriften unbedingt zu berücksichtigen! Man kann vieles mit einem kleinen angeschlossenen Verstärker und evtl. noch weiteren Hardwarezusätzen tun!
Es kommt nur darauf an, wie man das Ganze aufbaut und wo man anschließt.
Wir wollen uns im Moment darauf beschränken, einen reinen Mithörverstärker zu betrachten. Derartige Elektronikzusätze gibt es für ein paar Mark zu erstehen, so daß man nicht einmal viel selbst bauen muß. Diesen Verstärker schließt man signaleingangsseitig an den Pin D/4 des Kassettenports an. Wenn der Verstärker auch noch in der Lage ist, mit einer Versorgungsspannung von 5 V zu arbeiten, dann kann man diese getrost vom Pin B/2 beziehen, solange man nicht gerade mit einigen Watt Ausgangsleistung sein Umfeld mithören lassen will. Die gemeinsame Leitung (GND) ist an A/1 anzuschließen.
Das wäre eigentlich alles. Aber bestimmt wollen einige nun noch wissen, wie man Programme von einem zum anderen Computer mittels der angeführten Technik überträgt.

Programmübertragung

Um nun keine Vorschriften zu verletzen, nehmen wir an, die beiden Computer stehen in unmittelbarer Nähe zueinander.
Am „sendenden" Computer ist der Mithörverstärker angeschlossen und am „empfangenden" das Kassetteninterface mit einem angeschlossenen Mikrofon.
Damit sind die hardwaremäßigen Voraussetzungen bereits geschaffen. Nun am Empfänger „Load" eingeben und am Sender „Save". Nun müssen Sie nur noch warten, bis das Programm übertragen ist. Daß Sie während des Übertragungsvorganges Ihre vielleicht geliebte „Diskomusik" oder auch „Klassik" nicht ebenfalls in großer Lautstärke spielen sollten, versteht sich von selbst. Denn sonst könnte sich der empfangende C64 leicht „verhören".

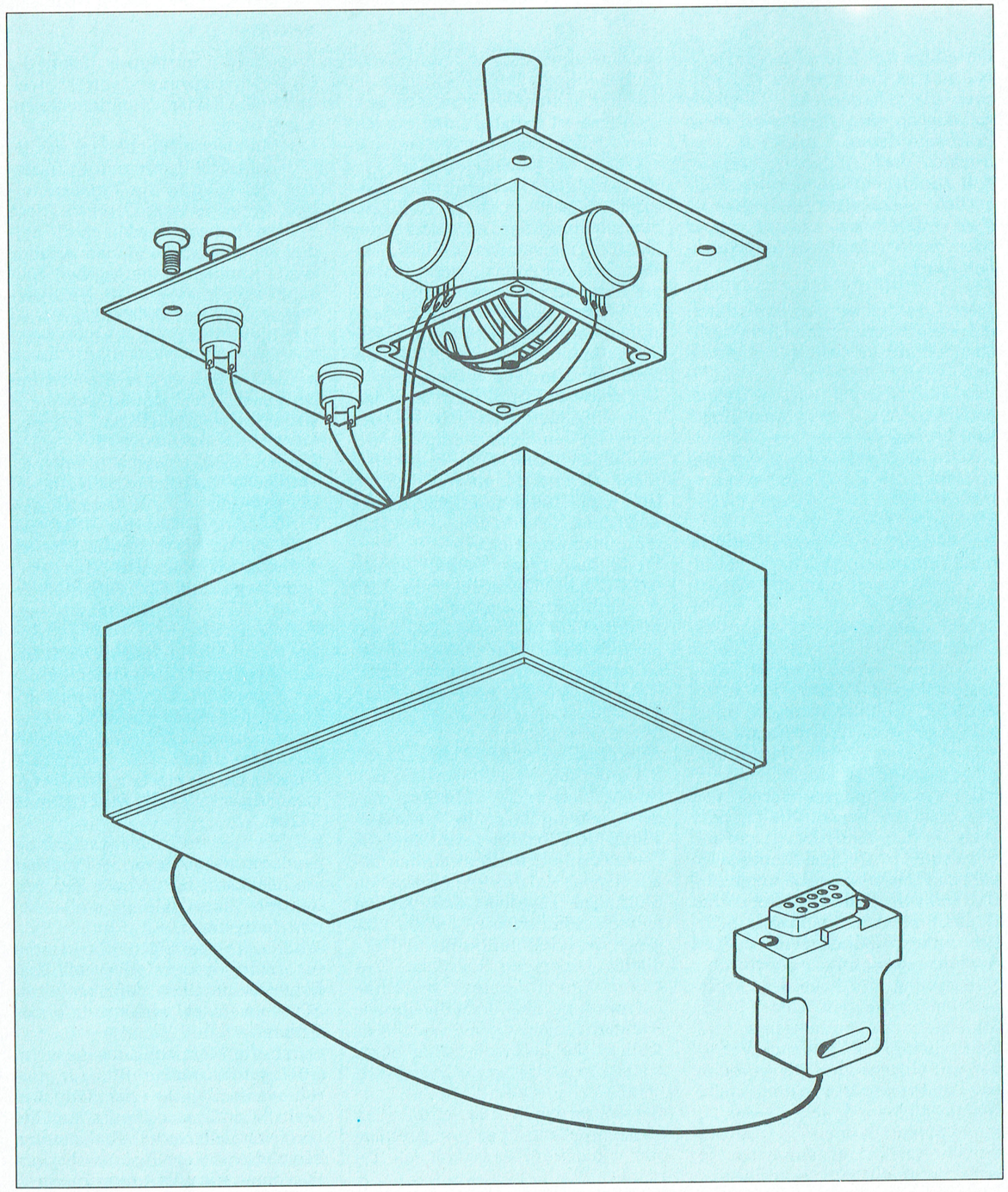


4. Analogjoystick

Für einige Spiel- und Malprogramme gibt es Touchtablets, mit welchen die erforderlichen Eingaben zu machen sind. Meist sind diese Zeichentableaus, preislich betrachtet, doch sehr hoch angesiedelt. Dies ist insbesondere dann ärgerlich, wenn dieses Eingabesystem irgendwann einmal defekt oder durch Abnutzung unbrauchbar wurde.

Ersetzt werden kann es dann durch die ebenfalls im Handel erhältlichen „Paddles", also den Drehreglern.
Man muß natürlich wissen, welche Sicherung in das Programm eingebaut ist, um das aus dem „Gesamtpaket" übriggebliebene Programm zu starten. Meist ist es nur erforderlich, die „Paddles" in eine gewisse Stellung zu bringen.
Der Umgang mit diesen ist aber nicht immer sehr praktisch, vor allem dann, wenn man mit Malprogrammen arbeitet.
Es gibt aber eine sehr preisgünstige Lösung, die evtl. ein ebenfalls sehr praktikables Arbeiten zuläßt. Diese Lösung liegt im Einsatz eines analogen Joysticks. Interessanterweise ist das Angebot derartiger Geräte sehr gering, aber es gibt für Bastler eine sehr preisgünstige Möglichkeit.
Für Fernsteueranlagen werden von einschlägigen Versandfirmen oder auch in den Modellbaugeschäften sogenannte „2-Kanal-Steuerknüppel" zu Preisen teilweise unter 10.– DM verkauft. Je nach Komfort und Qualität können die Preise natürlich unterschiedlich sein. Es gibt Ausführungen mit mechanischer Nullstellung, d.h., daß der „Knüppel" beim Loslassen wieder in die Mittelstellung zurückgeht, und wahrscheinlich gibt es auch Versionen mit eingebautem „Feuerknopf" usw. Im Prinzip ist für einen analogen Joystick, und dies ist der „2-Kanal-Steuerknüppel" ja, die einfachste Ausführung einsetzbar. Ja, es ist sogar die günstigste Lösung, da eine automatische mechanische Nullstellung ja nicht unbedingt der elektrischen entspricht. Zu berücksichtigen ist lediglich, daß die beiden Stellwiderstände (Potentiometer) einen Widerstandsbereich von ca. 200 Ω...250 kΩ haben sollen. Der niedrige Wert kann leicht durch Serienschaltung eines entsprechenden Zusatz-Widerstandes erreicht werden, falls der Anfangswert zu niedrig liegt. Eine Unterschreitung dieses Wertes könnte Ihnen der Computer leicht verübeln und einen Teil seines Innenlebens vorzeitig in den Halbleiterhimmel schicken.
Der Endwert aber sollte in etwa deshalb eingehalten werden, da man sonst mit dem System entweder die Endpunkte auf dem Bildschirm nicht erreicht oder aber sehr einseitig arbeitet und man durch geringste Bewegungen schon über den gesamten Bildschirm huscht.
Wenn man diese Punkte berücksichtigt hat, braucht man nur noch die Verbindungen mit dem Stecker herzustellen und zwei Taster als „Fire-Buttons" ein- oder anzubauen. Fertig ist der Analogjoystick. Der Rest dient nur noch der Schönheit, wie beispielsweise der Einbau in ein Gehäuse.
Allerdings ist zu beachten, daß die Belegung der Fire-Buttons bei den Grafiktablets nicht unbedingt mit der „Standardbelegung" übereinstimmen muß, das heißt, daß die Feuerknöpfe evtl. nicht an den üblicherweise dafür vorgesehenen Eingängen angeschlossen werden müssen, sondern u.U. an die Eingänge der „Richtungsschalter" der digital wirkenden Joysticks. Aus diesem Grunde können wir keine Patentlösung, aber doch die übliche Standardversion aufzeigen. Nun wollen Sie wahrscheinlich noch wissen, wie das ganze System arbeitet.
Der selbstgebaute Stick wird an den gleichen Controlport angeschlossen, wie das vorher verwendete Tablet.
Pin 7 des Controlports liefert die Versorgungsspannung von 5 V, welche an die beiden Potentiometer angelegt wird.
Der Mittelabgriff (Schleifer) des jeweiligen Potis liefert an die Eingänge 5 und 9 dann die Teilspannungen für die X- und Y-Achse. Diese Teilspannungen werden an die beiden im C64 vorhandenen Analog/Digitalwandler weitergegeben und verarbeitet. Das andere Ende der veränderlichen Widerstände wird mit dem Anschluß 8 (GND) verbunden.
Je nach Stellung des Steuerknüppels haben die Schleifer ebenfalls verschiedene Stellungen auf den Widerstandsbahnen.
Dies heißt, daß in Abhängigkeit hiervon auch unterschiedliche Spannungen an den Analogeingängen der beiden A/D-Wandler anstehen.
Diese werden dann in die entsprechenden Digitalwerte umgesetzt und entsprechen den „Koordinaten-Werten" für die Horizontal- und Vertikalposition.
Die Werte hierfür können dann aus den Registern 25 und/oder 26 des SID-(Sound-Interface-Device-)Bausteines ausgelesen und weiterverarbeitet werden. Die Grund- oder Basisadresse des SID ist 54272 ($d400), daraus ergeben sich die Registeradressen mit 54297 und 54298.
All dies aber und auch das nachfolgende ist, wenn Sie fertige Programme benutzen, für Sie weniger von Interesse, denn da liegt ja alles bereits fertig vor.
Wollen Sie aber eigene Programme für Analogjoysticks oder auch Grafiktablets schreiben, dann benötigen Sie auch noch weitere Informationen.
Durch die Mehrfachbelegungen im C64 werden einige Bits für verschiedene Zwecke benutzt, und deshalb muß man vor allem bei „professionelleren" Programmen beispielsweise die Tastatur abschalten, da es sonst zu unerwünschten

Effekten kommen kann, vor allem dann, wenn mit zwei Analogjoysticks (also mit vier! Potentiometern) gearbeitet werden soll, denn der C64 hat ja nur zwei A/D-Wandler. Wie dies ganz genau zu handhaben ist, werden wir in einer der normalen Ausgaben der Computer-Schau beschreiben.



5. Schalter: Geräteadresse- und Schreibschutz

Die Diskettenstation 1541 ermöglicht es, sowohl software- als auch hardwaremäßig mit einer anderen Geräteadresse als mit dem Standardwert von 8 betrieben zu werden. Dadurch ist es möglich, auch mehrere Floppystationen gleichzeitig am Bus betreiben zu können.

Die Softwarelösung mag zwar für viele Fälle ausreichend sein, ist unserer Meinung nach aber etwas umständlich zu handhaben. Praktikabler ist der Einbau von Adressenschaltern, die diese Umstellungen erlauben. Vorgesehen ist es in den Floppydrives, daß dies geschehen kann. Aber ein Schalter ist nicht eingebaut, so daß man dies selbst nachholen muß, wenn man die Geräteadressen schaltbar haben will.
Im Gerät befinden sich zu diesem Zwecke zwei sogenannte Leiterbahnschalter. Es handelt sich hierbei natürlich um keine Schalter im üblichen Sinne des Wortes, sondern um größere Kontaktpunkte auf der Platine, die durch dünnere Kontaktbahnen verbunden sind. Wenn nun diese dünnen Bahnen aufgetrennt werden, ändern sich, ganz allgemein ausgesagt, die Geräteadressen. Nun aber etwas genauer.
Wenn man den Gehäusedeckel der 1541 öffnet und einen Blick in das Innenleben wirft, kann man neben den mechanischen Teilen auch sehr leicht einen elektronischen Teil in Form der Hauptplatine erkennen. Auf dieser Platine sind die beiden für Sie interessanten Platinenschalter. Um Sie zu finden, stellen Sie bitte die geöffnete Diskettenstation so vor sich hin, daß sich der Diskenteneinschubschacht vorne und die Anschlußbuchsen sich – von Ihnen aus betrachtet – hinten befinden. Daß Sie die 1541 dabei nicht auf den Kopf stellen sollen, versteht sich von selbst! Nun sehen Sie im vorderen linken Drittel der Platine an der linken, also der vorderen Ekke die Befestigungsschraube, rechts daneben können Sie sechs gleichartige Bauelemente erkennen (der Platinenaufdruck dort: CR18 – CR13), bei diesen Bauteilen handelt es sich um Dioden. Rechts neben diesen Dioden folgt dann ein Bauelement mit drei Anschlußbeinen (Transistor), dessen Bezeichnung ist: Q7B.
Lassen Sie nun Ihren Blick von vorne nach hinten gleiten, so sehen Sie noch weitere drei dieser Transistoren. Zwischen dem (von vorne gezählt) dritten. Transistor (Q5B) und dem vierten. (Q4B) können Sie nun die beiden „Leiterbahnschalter" erkennen.
Jeder dieser beiden Schalter besteht aus zwei Kreisabschnitten, die mittels einer dünneren Leiterbahn verbunden sind.
Die eine Seite dieser Kreisabschnitte ist mit Masse verbunden. Dies bedeutet, daß im Originalzustand beide Schalter den logischen Wert 0 haben. Unterbricht man nun diese beiden Schalterhälften derart, daß man die Verbindungsstege auftrennt (Rasierklinge, scharfes Messer), so führen die Schalter den logischen Wert 1.
Da es sich um zwei Schalter handelt, kann man damit vier verschiedene Kombinationen von logischen Zuständen erreichen.
Liegen beide Schalter auf Masse, so ist die Geräteadresse 8 eingestellt.
Den vorderen bezeichnen wir nun willkürlich mit S1, den hinteren mit S2.
Nun gleich eine kleine Tabelle, aus der Sie die Geräteadressen in Abhängigkeit von der „Schalterstellung" ablesen können:

S 1	S 2	Geräteadresse
0	0	8
1	0	9
0	1	10
1	1	11

Diese kleine Übersicht sagt Ihnen nun bereits alles. 0 bedeutet geschlossener Kontakt, 1 offener.
Ob Sie nun einen kleinen Dip-Schalter, einen oder zwei Kippschalter, einen Drehschalter oder was sonst auch immer in das Gerät selbst oder an das Gerät bauen oder auch nur entsprechende Durchtrennungen der Platinenschalter vornehmen, bleibt Ihrem persönlichen Wunsch und Geschmack bzw. Ihren Erfordernissen überlassen.
Wo und wie man's macht, haben wir Ihnen aufgezeigt.
Wichtig für den praktischen Betrieb ist allerdings, daß die entsprechend gewünschte Adresse vor dem Einschalten ausgewählt wird. Beziehungsweise, daß dann, wenn man die Adresse hardwaremäßig umschalten will, die Floppy jedesmal aus-(und nach der Änderung wieder ein-)geschaltet werden muß.

Aufhebung des Schreibschutzes

Wie Sie wissen, wird ein Schreibschutz dadurch erreicht, daß an den Disketten der „Schreibschutzaus-

schnitt“ mit einer lichtundurchlässigen Folie, Papier usw. abgedeckt wird. Es gibt zwei Hauptgründe dafür, den Schreibschutz aufhebbar zu machen.

Der erste und allgemein häufigste ist, daß auf eine „schreibgeschützte“ Diskette geschrieben werden soll. Dies heißt, daß die „Abdekkung“ entfernt werden muß. Will man dann die Scheibe wieder schützen, heißt es wieder neu abdecken usw. Eine manchmal doch etwas ärgerliche Prozedur, vor allem dann, wenn im Augenblick gerade wieder kein geeigneter Abdeckstreifen in greifbarer Nähe ist. Der Autor (und nun darf ruhig geschmunzelt werden) arbeitete jahrelang bereits mit der Floppy 3040 und war es gewohnt, diesen Schutz mit kleinen Tesastreifen zu realisieren. Nichts lag also näher, als dies auch bei den Disketten für die 1541 zu tun. Ein Datenverlust größeren Umfanges war die Folge.

Im Gegensatz nämlich zur 3040 (kleine Mikroschalter übernehmen die Abfrage) arbeitet die Schreibschutzerkennung bei der 1541 mit Optoelektronik.

Wenn Sie nun zu Ende geschmunzelt haben, fahren wir mit ernsthafteren Dingen fort, nämlich mit dem zweiten Grund.

Wie Sie vielleicht schon wissen, können viele Disketten (auch einseitige!) beidseitig verwendet werden. Die Hersteller übernehmen bei einseitig deklarierten zwar keine Garantie dafür, daß dies möglich ist – ja, sie lehnen dies sogar ab –, aber bei einigen Fabrikaten ist dies fast 100 %ig möglich. Es wurden sogar Testprogramme geschrieben, die überprüfen, ob sich die jeweilige Diskette dazu eignet. Mag es für professionelle Anwender wichtig und richtig sein, aus Gründen der Datensicherheit „Einseitige“ auch nur so zu verwenden, für den Hausgebrauch ist dies – sofern es sich nicht um wichtige Datendisketten handelt – nicht so vorrangig. Bei den Freaks hatte sich diese Tatsache sehr schnell herumgesprochen und so wurden mittels mehr oder weniger geeigneter Instrumente bzw. Werkzeuge die Disketten so präpariert, daß sie beidseitig verwendbar waren. Abgesehen davon, daß diese Einschnitte nicht immer die (vom ästhetischen Standpunkt her betrachtet) saubersten waren, besteht auch leicht die Gefahr, daß man mit ungeeignetem Werkzeug die Disketten auch noch beschädigt. Deshalb wird z. B. auch Werkzeug für diese Manipulation angeboten.

Nichts über den Marktpreis dieser Tools, bzw. auch kein Urteil darüber, ob dieser gerechtfertigt ist oder nicht. Man kann das Ganze viel billiger durch einen einmaligen Eingriff in die 1541 tun.

Die beiden Gründe also für den Einbau dürften klar sein. Nun aber auch wieder erst dazu, wie dies im Gerät selbst funktioniert. Bei den ganzen Beschreibungen haben wir bisher immer versucht, die Dinge so aufzuzeigen, daß auch ein Nichtelektroniker klarkommt und dies wollen wir auch hier wieder so halten.

Im vorderen Teil der linken Diskettenführungsschiene befindet sich ein optoelektronisches System, welches aus einer Fotodiode (Lichtsender) und einem Fototransistor (Lichtempfänger) besteht. Wurde bei einer Diskette kein Schreibschutz geklebt, so kann das emittierte (ausgesandte) Licht von der Diode zum Transistor gelangen. Bei geklebtem Schutz wird der Lichtstrahl daran gehindert.

Dadurch ist nun schon klar, was man eigentlich zu tun hat, wenn man dieses System „austricksen“ will. Man muß der Folge-Elektronik nur mitteilen, daß der Fototransistor Licht „sieht“. Im Klartext heißt dies, daß man parallel zum Transistor einen zuschaltbaren „Bypaß“ legen muß. Um es technisch korrekt durchzuführen, müßte man einen Widerstand, der dem Wert des durchgeschalteten Transistors entspricht (ca. 250...280 Ω), zuschaltbar über den „Transistorschalter“ legen. Der Autor hat allerdings, ohne daß Probleme irgendwelcher Art bisher aufgetreten sind, dies ohne den Widerstand realisiert.

Wer keinen „Schutzwiderstand“ einbaut, ist selbst schuld, wenn dadurch ein Schaden auftritt.

Dies kann also jeder halten, wie er will.

Andererseits aber hat der Autor bei seiner Floppy gleich noch eine optische Anzeige, daß der Schutz abgeschaltet ist, integriert. Um möglichst wenig am Gehäuse bohren zu müssen, um aber andererseits mitgeteilt zu bekommen, daß „Schreibgefahr“ besteht und außerdem nicht aus Versehen umgeschaltet wird, erfolgte der Schaltereinbau in den hinteren Gehäuseunterteil und die normale Betriebsanzeige-LED (Light Emitting Diode) wurde gegen eine Doppel-LED ausgetauscht.

Nun zu den Details: Benötigt wurden

1 Schalter (2 × Ein)
1 Doppel-LED (mit Fassung)
1 Widerstand 220 Ω

Hierbei ist der Widerstand, der den durchgeschalteten Transistor simuliert, nicht berücksichtigt.

An dem zum Stecker P6 (Platinenaufdruck) führenden Anschlußdraht 12 (orange) wurde ein weiterer Draht, der zum Schalter führt, angelötet. Der andere Teil dieses Schaltersegments wurde mit Masse verbunden (z. B. Anschluß 13 an der Steckerleiste P6). Der grüne Teil der Doppel-LED wurde wie im Originalzustand verdrahtet und der Rotteil über Schaltsegment 2 und dem Vorwiderstand von 220 Ω auf +5 V gelegt. Das war alles und arbeitet nun seit Monaten einwandfrei. Normalerweise leuchtet die Betriebsanzeige-LED nun wie bisher grün; wird umgeschaltet, dann bewirkt der hinzukommende Rotteil eine orange Anzeige.

Vorsicht!

Was auf der einen Seite von Vorteil, kann andererseits von Nachteil sein. Das DOS (Disk Operating System) erkennt nun nämlich keinen Diskettenwechsel mehr, wenn der „Überlist“-Schalter aktiv ist. Dies kann bewirken, daß Fehler in bezug auf die Directory und ID auftreten können, da ein Wechsel nicht erkannt wird. Dem könnte man durch einen von der Diskettenklappe betätigten weiteren Schalter entgegenwirken, aber vielleicht will der eine oder andere dies gar nicht haben, wenn nämlich Disketten manipuliert werden sollen!

Hacker – ein Spiel nach der Wirklichkeit

Die Hacker sind in aller Munde, die bösen Computerpiraten, die sich in fremde Computernetze einschleichen und dort Daten nassauern. Die USA sind ihr Eldorado, aber auch in der Bundesrepublik Deutschland treiben sie schon ihr Computer-Unwesen.

Das hier beschriebene Programm für den Commodore 64 gibt einen Einblick in die Arbeitsweise und Gefahren im Leben eines „Hackers“. Dabei geht es darum, daß man mit seinem Computer über die Telefonleitung in eine fremde Datenbank eindringt.

Zuerst einmal muß der Spieler den Anschluß, also die Telefonnummer der fremden Datenbank finden. Er läßt seinen C 64 wählen, und das kann unter Umständen lange dauern, denn die gesuchte Nummer muß aus 9000 Anschlüssen herausgefunden werden. Und damit das Suchen nicht zu leicht ist, wechselt die Nummer nach einer gewissen Zeit wieder.

Falls der Computer nun die Nummer gefunden hat und sich das fremde Datenzentrum meldet, muß noch eine Hürde genommen werden – man muß das Codewort finden, durch das man erst Zugang zu den Daten bekommt. Das Codewort entstammt der Welt des Computers, ist also nicht so schwer zu finden. Doch Vorsicht, man hat nur drei Versuche, dann hat man den Eindringling entdeckt. Und das bedeutet die Verfolgung durch die Polizei – und im Spiel den völligen Zusammenbruch des Hackerprogramms, das dann erst wieder neu geladen werden muß.

Hat man aber Glück gehabt und das Codewort gefunden (das natürlich auch bei jedem Spiel wieder wechselt), dann kann der Spieler die Informationen aus der gefundenen Datenbank abrufen.

Natürlich wird man weder die Telefonnummern noch die Codeworte in diesem Listing finden, das würde die Sache ja zu einfach machen. Man versuche auch nicht, sich das Paßwort durch eine zusätzliche Printzeile auf dem Bildschirm anzeigen zu lassen; das Programm läuft dann nämlich einfach nicht mehr.

Michael Schütz

Listing: Hacker

```
100 REM START
110 PRINTCHR$(14):POKE53280,8:POKE53281,6
120 PRINT"■■■■■■■■■■■■■■■■■■■■■■■■■■■■■OMPUTER♥CHAU
130 FORJ=15TO0STEP-1:FORI=412TO440:POKE55296+I,J
140 NEXTI:NEXTJ
150 REM (C) MICHAEL SCHUETZ,WUPPERTAL 1984
160 PRINT"■";
170 PP$="■■■■■■■■■■■■■■■■■■■
```

```
370 POKE 204,1:PRINT"■";
380 SI=54272:FL=SI:FH=SI+1:TL=SI+2:TH=SI+3:WE=SI+4
390 AN=SI+5:HA=SI+6:LA=SI+24
400 PRINT"■"
410 GOSUB 2360
420 PRINT"■"
430 PRINT"IN IHREM OERTLICHEN FERNSPRECHNETZ"
440 PRINT"BEFINDEN SICH AUCH DATENVERBINDUNGEN.■"
450 PRINT"SIE KOENNEN SICH AUF EINE LEITUNG"
460 PRINT"SCHALTEN, WENN SIE EINE"
470 PRINT"VIERSTELLIGE RUFNUMMER WAEHLEN.■"
480 PRINT"SIE KOENNEN AUCH EINEN RUFNUMMERN-"
490 PRINT"BEREICH (MAX.1000) ABSUCHEN LASSEN,"
500 PRINT"WENN SIE DIE ERSTE UND LETZTE NUMMER"
510 PRINT"ANGEBEN."
520 PRINT"FINDET DER COMPUTER EINE DATENLEITUNG,"
530 PRINT"AUF DER GEARBEITET WIRD, HABEN SIE"
540 PRINT"DIE MOEGLICHKEIT, SICH IN DAS FREMDE"
550 PRINT"COMPUTERSYSTEM EINZUSCHALTEN, WENN SIE"
560 PRINT"DAS RICHTIGE PASSWORT EINGEBEN."
570 PRINT"■WENN'S LOSGEHEN SOLL, DRUECKEN SIE"
580 PRINT"DIE ■FUNKTIONSTASTE F1"
590 FORI=1TO999:FORJ=1TO3
600 POKE55296+I,J
610 GET T$:IF T$="" THEN 630
620 IFT$=CHR$(133)THEN640
630 NEXTJ:NEXTI:GOTO590
640 NR%=INT(10000*RND(1))
650 IF NR%<1000 THEN 640
660 DU%=INT(1000*RND(1))+1000
670 PRINT "■"
680 FOR I=1 TO 63:READ A:NEXT
690 GOSUB 2200
700 PRINT"■";
710 INPUT"GEBEN SIE EINE RUFNUMMER EIN:";R1$:R1=VAL(R1$)
720 IFR1<1000ORR1>9999THENPRINT"■■■■";:PRINTF2$:GOSUB2760:GOTO700
730 RL=R1
740 PRINT"■WOLLEN SIE MEHRERE RUFNUMMERN          ABSUCHEN LASSEN? (J/N)"
750 GET T$:IF T$="" THEN 750
760 IF T$="J" THEN 790
770 IF T$="N" THEN 860
780 GOTO 750
790 PRINT "■■■■■■■";
800 INPUT"GEBEN SIE DIE LETZTE NUMMER EIN:";RL$:RL=VAL(RL$)
810 IFRL<R1ORRL>9999ORRL>R1+1000THENPRINTF3$:FORI=1TO5000:NEXTI:RUN100
820 PRINT"■■■■■■■■";
830 PRINT"ES WIRD VON ";R1;"BIS ";RL;" GEWAEHLT."
840 PRINT"■BITTE WARTEN"
850 GOTO890
860 PRINT"■■■■■■■";
870 PRINT"ES WIRD ";R1;" GEWAEHLT."
880 PRINT:PRINT"BITTE WARTEN"
890 FORI=R1TORL
900 IF I=NR% THEN 1040
910 DU%=DU%-1
920 IF DU%=0 THEN NR%=0
930 NEXTI
940 FORI=1TO52:READA:NEXT
950 GOSUB2530
960 PRINT"KEINE DATENLEITUNG GEFUNDEN■"
970 PRINT"NEUWAHL? (J/N)"
980 GET T$:IF T$="" THEN 980
990 IF T$="J" THEN 1020
```



```
1000 IF T$="N" THEN END
1010 GOTO 980
1020 IF NR%=0 THEN 640
1030 GOTO 670
1040 REM LEITUNG GEFUNDEN
1050 FOR I=1 TO 57:READ A:NEXT
1060 GOSUB 2530
1070 PRINT "■"
1080 FOR I=1 TO 181:READ A:NEXT
1090 GOSUB 2200
1100 PRINT"■■■■";TAB(21);CHR$(156);
1110 FOR I=1 TO 13:PRINT"*";:NEXT:PRINT"*"
1120 PRINT TAB(21);"*";SPC(12);"*"
1130 PRINT TAB(21);"*";CHR$(30);" MS-RECHNER ";CHR$(156);"*"
1140 PRINT TAB(21);"*";SPC(12);"*"
1150 PRINT TAB(21);"*";SPC(12);"*"
1160 PRINT TAB(21);"*";CHR$(30);" WUPPERTAL  ";CHR$(156);"*"
1170 PRINT TAB(21);"*";SPC(12);"*"
1180 PRINT TAB(21);"*";SPC(12);"*":PRINT TAB(21);
1190 FOR I=1 TO 14:PRINT"*";:NEXT:PRINT
1200 PRINT"■";:I=0
1210 PRINT LEFT$(PP$,15);
1220 PRINT"BITTE PASSWORT EINGEBEN:"
1230 INPUT PA$
1240 IF PA$=PW$ THEN 1430
1250 PRINT"FALSCHES PASSWORT"
1260 I=I+1:IF I=3 THEN 1280
1270 GOTO 1210
1280 PRINT"■":FORI=1TO40:PRINTTAB(10)" ■ A L A R M   !!!! ":NEXTI
1290 PRINT"■■■■■■■■■■■■SIE VERSUCHEN, WIDERRECHTLICH"
1300 PRINT"IN UNSEREN COMPUTER EINZUDRINGEN !"
1310 PRINT"SIE HABEN SICH STRAFBAR GEMACHT."
1320 PRINT"DIE OERTLICHE KRIMINALPOLIZEI IST"
1330 PRINT"VERSTAENDIGT. IHR PROGRAMM WIRD"
1340 PRINT"JETZT ZERSTOERT !■■"
1350 PRINT"■■■SIE MUESSEN LEIDER WIEDER NEU LADEN !!)
1360 FORY=1TO100
1370 FORI=1TO10
1380 POKE53281,I:POKE53280,I+1
1390 NEXTI:NEXTY
1400 POKE53281,1
1410 SYS58260
1420 POKE(PEEK(43)+256*PEEK(44))+3,128:END
1430 REM LOG IN
1440 LG$="■■                                    "
1450 L1$="■■                                        "
1460 L2$="■■ ■                                      ■ "
1470 L3$="■■ ■ ■                                    ■ ■ ."
1480 PRINT"■"
1490 PRINT TAB(4);L1$
1500 PRINT TAB(4);L2$
1510 FOR I=1 TO 18:PRINT TAB(4);L3$:NEXT
1520 PRINTTAB(4);L2$:PRINTTAB(4);L1$
1530 PRINT"■";
1540 PRINT"■■■■■■■";
1550 PRINT TAB(6);"LOGISCHE VERBINDUNG"
1560 PRINT TAB(6);"HERGESTELLT !":PRINT
1570 PRINT TAB(6);"DATEIVERZEICHNIS: ":PRINT
1580 PRINT TAB(6);"F1:UMWELTSCHUTZ "
1590 PRINT TAB(6);"F2:PARTEISPENDEN "
1600 PRINT TAB(6);"F3:GEHEIMGLEICHUNGEN "
1610 PRINTTAB(6);"F4:NUMMERN  ":PRINT"■■■"
1620 PRINT TAB(6);"F8:VERBINDUNGSABBRUCH "
```

```
1630 GET A$:IF A$="" THEN 1630
1640 IF A$=CHR$(133) THEN 1700
1650 IF A$=CHR$(137) THEN 1890
1660 IF A$=CHR$(134) THEN 1940
1670 IF A$=CHR$(138) THEN 2030
1680 IF A$=CHR$(140) THEN 2120
1690 GOTO 1630
1700 GOSUB 2170
1710 PRINT TAB(6);"GEFAEHRDETE GEBIETE: ":PRINT
1720 PRINT TAB(14);"■  ▬▬          "
1730 PRINT TAB(14);"■  ▄  HH● ▙     "
1740 PRINT TAB(14);"■  ▄     ▟     "
1750 PRINT TAB(14);"■  ▄▖    ▙     "
1760 PRINT TAB(14);"■  ▄  ▗  ▄     "
1770 PRINT TAB(14);"■  ▄●K ▗  ▀     "
1780 PRINT TAB(14);"■▀▀▗  ▘    ▄     "
1790 PRINT TAB(14);"■▀▀▗  ▘    ▄     "
1800 PRINT TAB(14);"■ ▗ FF●  ▀▀▙    "
1810 PRINT TAB(14);"■ ▗         ▗ ▙  "
1820 PRINT TAB(14);"■ ▀▖        ▗  ▌"
1830 PRINT TAB(14);"■  ▝▀▙ ▘  M● ▗  ▄ "
1840 PRINT TAB(14);"■   ▄▀ ▘      ▄ "
1850 PRINT TAB(14);"■   ▄▄  ▘      ▟ "
1860 PRINT TAB(14);"               "
1870 PRINT TAB(6);"F8";:POKE 204,0
1880 GOTO 2130
1890 GOSUB 2170
1900 PRINTTAB(6);"SPENDENKONTEN: "
1910 PRINTTAB(6);"■MICHAEL SCHUETZ "
1920 PRINT TAB(6);"5600 WUPPERTAL 1 "
1930 PRINT"▄▄▄▄▄▄":GOTO 1870
1940 GOSUB 2170
1950 PRINTTAB(6);"GEHEIMGLEICHUNGEN: "
1960 PRINTTAB(6);"■U = I * R "
1970 PRINTTAB(6);"■I = U / R "
1980 PRINTTAB(6);"■R = U / I "
1990 PRINTTAB(6);"■I = STROM "
2000 PRINTTAB(6);"■U = SPANNUNG "
2010 PRINTTAB(6);"■R = WIDERSTAND "
2020 PRINT"▄▄▄":GOTO1870
2030 GOSUB 2170
2040 PRINTTAB(6);"NUMMERN : ":PRINT
2050 PRINTTAB(6);"AUSKUNFT       0118"
2060 PRINTTAB(6);"LOTTERIE     011607"
2070 PRINTTAB(6);"GESUNDHEIT   011602"
2080 PRINTTAB(6);"REZEPTE       01167"
2090 PRINTTAB(6);"TIPS         011606"
2100 PRINTTAB(6);"WETTER        01164"
2110 PRINT"▄▄▄▄▄▄▄":GOTO1870
2120 PRINT"▄▄▄▄▄▄▄▄▄VERBINDUNGSABBRUCH■":END
2130 GET A$:IF A$="" THEN 2130
2140 IF A$= CHR$(140) THEN 2160
2150 GOTO 2130
2160 GOSUB 2170:GOTO 1540
2170 PRINT"▄▄▄▄";:POKE 204,1
2180 FOR I=1 TO 18:PRINT TAB(6);LG$:NEXT
2190 PRINT"▄▄▄▄";:RETURN
2200 REM GRAFIK
2210 READ A:PRINT LEFT$(PP$,A):READ A:PS=A
2220 PRINT"■";
2230 READ A:PRINT CHR$(A);
2240 READ A:PRINT TAB(PS+A);
2250 READ A
```



```
2260 IF A=-1 THEN PRINT:GOTO 2240
2270 IF A=-2 THEN 2320
2280 IF A=-3 THEN 2310
2290 IF A=-4 THEN 2350
2300 PRINT CHR$(A);:GOTO 2250
2310 READA:READB:FORI=1TOA:PRINTCHR$(B);:NEXTI:GOTO2250
2320 RESTORE:FORI=1TO63:READA:NEXTI:READA
2330 PRINT LEFT$(PP$,A+5);:PRINT"■";
2340 READ A:PRINT TAB(A+8);"?"
2350 PRINT"■":RESTORE:RETURN
2360 REM ZUWEISUNGSTEIL
2370 PW$=""
2380 IX%=INT(10*RND(0))+1
2390 FOR I=1 TO 10
2400 READ A
2410 IFIX%<>ITHEN2430
2420 A1=A
2430 NEXT I
2440 AH=INT(A1/100):AZ=INT((A1-100*AH)/10):AE=A1-100*AH-10*AZ
2450 FOR I=1 TO AH
2460 READ A
2470 NEXT I
2480 FOR I=1 TO AZ
2490 READ A
2500 PW$=PW$+CHR$(A-AE)
2510 NEXT I
2520 RESTORE:RETURN
2530 REM TELEFONTON
2540 READ A:IF A=-1 THEN RESTORE:RETURN
2550 ON A GOTO 2560,2570,2590,2600,2610,2620
2560 X7=0:X8=0:X9=100:GOTO 2630
2570 X0=85:X1=0:X2=0:X3=8:X4=65:X5=0:X6=240:X7=1000:X8=500:X9=6
2580 GOTO 2660
2590 X7=800:X8=2000:X9=6:GOTO 2630
2600 X7=200:X8=200:X9=15:GOTO 2630
2610 X7=200:X8=500:X9=6:GOTO 2650
2620 X7=0:X8=0:X9=40:GOTO 2650
2630 X0=69:X1=29:X2=0:X3=0:X4=17:X5=0:X6=240
2640 GOTO 2660
2650 X0=134:X1=66:X2=0:X3=0:X4=17:X5=0:X6=240
2660 POKE LA,15
2670 POKE FL,X0:POKE FH,X1:POKE AN,X5:POKE HA,X6
2680 POKE TL,X2:POKE TH,X3
2690 X9=X9-1:IF X9>0 THEN 2710
2700 POKE1802,63:POKEWE,0:GOTO2540
2710 POKE WE,X4:IF X7=0 THEN 2730
2720 FORI=1TOX7:NEXTI
2730 IF X8=0 THEN 2690
2740 POKEWE,0:FORI=1TOX8:NEXTI:POKE1802,33
2750 GOTO 2690
2760 PRINT"■":PRINTF1$
2770 FORF=1TO10
2780 PRINT"▄▄▄";F1$
2790 PRINT"▄▄";F1$
2800 NEXTF
2810 FORF=1TO500:NEXTF:PRINT"■":PRINTBL$:RETURN
2820 REM DATEN PASSWORT
2830 DATA 134,453,941,1335,1646,2041,2442,2864,3441,3837
2840 DATA 34,71,84,89,75,68,79,79,82,72,66,84,85
2850 DATA72,59,57,90,75,89,90,68,80,69,70,74,71,78,82
2860 DATA 76,69,71,79,73,86,67,90,85,70,73,80,91,34
2870 REM DATEN TON (+52)
2880 DATA 1,2,3,4,-1,1,2,3,5,6,-1
```

```
2890 REM DATEN BILD TELEFON ( +63)
2900 DATA 15,10,158,2,175,162,18,-3,9,32,146,162,175,-1
2910 DATA 0,162,18,-3,15,32,146,162,-1
2920 DATA 0,18,-3,5,32,162,-3,5,32,162,-3,5,32,146,-1
2930 DATA 0,18,32,32,175,146,32,32,18,-3,7,32,146,32,32,18,175,32,32,146,-1
2940 DATA 4,18,32,32,146,183,32,32,32,183,18,32,32,146,-1
2950 DATA 4,18,32,146,162,32,32,32,32,32,162,18,32,146,-1
2960 DATA 3,18,32,32,32,32,184,184,184,32,32,32,32,146,-1
2970 DATA 3,18,-3,11,32,146,-1
2980 DATA 2,18,-3,13,32,146,-2
2990 REM DATEN BILD RZ ( +181)
3000 DATA 2,5,158,3,18,169,-3,8,32,146,-1
3010 DATA 2,18,169,-3,9,32,146,-1
3020 DATA 2,144,18,172,-3,6,162,187,158,32,32,146,-1
3030 DATA 2,144,161,18,30,-3,6,32,144,161,158,32,32,146,-1
3040 DATA 2,144,161,18,30,-3,6,32,144,161,158,32,32,146,-1
3050 DATA 2,144,161,18,30,-3,6,32,144,161,158,32,32,146,-1
3060 DATA 2,144,18,188,146,-3,6,162,18,190,158,32,146,169,-1
3070 DATA 1,18,169,-3,7,113,146,169,169,-1
3080 DATA 0,18,169,-3,7,113,146,169,-1
3090 DATA 0,18,-3,8,162,146,-4
3100 REM DATEN TITELBILD ( +299)
3110 DATA 0,2,159,15,18,169,-3,18,32,-1
3120 DATA 14,18,169,-3,19,32,-1
3130 DATA 13,18,169,-3,20,32,-1
3140 DATA 12,18,169,-3,6,32,158,169,32,32,159,32,32,158,169,32,32,159,32,32
3150 DATA 158,169,32,32,159,32,32,-1
3160 DATA 11,18,169,32,158,169,146,169,18,159,32,32,32,32,144,113,113,158,32,159
3170 DATA 32,32,144,113,113,158,32,159,32,32,144,113,113,158,32,159,32,32,-1
3180 DATA 10,18,169,32,158,169,146,169,144,18,191,159,146,169,32,32,32,18,30,165
3190 DATA 167,158,32,31,32,32,30,165,167,158,32,31,32,32,30,165,167,158,32,-1
3200 DATA 9,18,159,169,32,32,158,32,144,191,30,169,146,32,32,32,32,18,108,186
3210 DATA 146,158,169,32,32,18,30,108,186,146,158,169,32,32,18,30,108,186,146
3220 DATA 158,169,-1
3230 DATA 8,18,159,169,32,32,32,158,32,30,169,146,169,32,18,158,169,32,32,31
3240 DATA 32,32,158,169,32,32,-1
3250 DATA 7,18,159,169,32,32,32,32,158,32,30,146,169,32,167,18,111,112,158,32
3260 DATA 146,32,167,30,18,111,112,158,32,31,32,32,158,169,32,32,146,169,-1
3270 DATA 6,18,159,169,32,32,32,32,146,169,32,32,32,167,18,30,108,186,158,146
3280 DATA 169,165,167,30,18,108,186,146,158,169,165,18,169,32,32,146,169
3290 DATA 144,18,191,-1
3300 DATA 6,18,28,-3,7,32,188,146,32,158,167,184,184,165,32,167,184,184,165,32
3310 DATA 18,158,32,32,32,144,191,18,30,169,-1
3320 DATA 6,18,28,-3,8,32,188,31,-3,10,32,158,32,32,32,30,169,32,-1
3330 DATA 6,18,28,-3,9,32,188,31,-3,9,32,158,32,32,32,30,32,146,169,-1
3340 DATA 6,18,28,-3,12,32,188,31,-3,6,32,158,32,32,32,30,146,169,-1
3350 DATA 4,18,158,169,-3,8,32,28,-3,6,32,188,184,-1
3360 DATA 3,18,158,169,-3,9,32,28,-3,21,162,-1
3370 DATA 3,144,18,172,-3,6,162,187,158,32,32,-1
3380 DATA 3,144,161,18,30,-3,6,32,144,161,158,32,32,-1
3390 DATA 3,144,161,18,30,-3,6,32,144,161,158,32,32,-1
3400 DATA 3,144,161,18,30,-3,6,32,144,161,158,32,32,-1
3410 DATA 3,144,18,188,146,-3,6,162,18,190,158,32,146,169,-1
3420 DATA 2,18,169,-3,7,113,146,169,169,-1
3430 DATA 1,18,169,-3,7,113,146,169,-1
3440 DATA 1,18,-3,8,162,146,-4
3450 REM **********************

READY.
```


Alles über Mikro-Computer

mc-Sonderheft **NEU**
Mikrocomputer Schritt für Schritt

Direkt ins Zentrum der Mikroelektronik führt das Sonderheft „Mikrocomputer Schritt für Schritt". Mit diesem Begleitheft zur Fernsehreihe des NDR kann jeder einen Computer selbst aufbauen und verstehen lernen.

Ausschließlich neue Beiträge. Für Anfänger und Fortgeschrittene.

120 Seiten, 28 DM

mc-Sonderheft **NEU**
DAS APPLE-SONDERHEFT

In diesem Sonderheft der mc finden Sie zahlreiche Programme für den Apple-II und seine „kompatiblen" Abkömmlinge, aber auch Beiträge über die Arbeitsweise des Applesoft-Interpreters und des Disketten-Betriebssystems DOS 3.3. Die Beiträge stammen überwiegend aus inzwischen größtenteils nicht mehr lieferbaren mc-Heften – und sind eben doch aktuell geblieben.

Lieferbar ab Anfang Oktober.

Für Anfänger und Fortgeschrittene.

96 Seiten, 14 DM

Das
mc-MODEM-Sonderheft

behandelt Hard- und Software zur Datenübertragung per Telefon – Schaltungstechnik von Modems. Übertragungs-Software, Grundlagen der Datenfernübertragung, Schnittstellen an Computern. Übertragungsprotokolle, öffentlich zugängliche Datenbanken.

Überwiegend Aufsätze aus mc. Für Profis und Hobbyisten.

64 Seiten, 16 DM.

mc-Sonderheft
Das EMUF-Sonderheft

Aufbau, Programmierung und Anwendung eines **E**inplatinen-**M**ikrocomputers für **u**niverselle **F**estprogramm-Anwendung.

3., überarbeitete Auflage mit 24 zusätzlichen Seiten.

Überwiegend neue Beiträge.

Für Anfänger und Fortgeschrittene, 6502-Assembler-Programmierung.

88 Seiten, 18 DM

FUNKSCHAU-Sonderheft
Klartext für den ZX 81

Mit diesem Sonderheft erlernt sich Maschinensprache so leicht wie nie zuvor! In kleine überschaubare Portionen aufgeteilt wird ein fesselnder Lehrstoff geboten, weil der Leser zahlreiche Beispiele selbst auf seinem ZX 81 nachvollziehen kann. Die nötigen Vorkenntnisse vermittelt das dem Computer beiliegende Original-Sinclair-Handbuch.

Ergänzte Zusammenfassung aller 27 Folgen.

Für Anfänger und Hobbyisten.

64 Seiten, 14 DM

FUNKSCHAU-Sonderheft
Zaubern mit dem ZX 81

Wer dieses Heft liest, entdeckt völlig neue Seiten des ZX 81 und kann ihn besser nutzen, z. B. wird das Programmieren in Maschinensprache durch einen kompletten Lehrgang leichtverständlich.

Besprechungen käuflich erwerbbarer Software erleichtern die Auswahl.

Für Mikrocomputer-Hobbyisten und -Einsteiger.

Ein Teil der Beiträge wurde bereits in der FUNKSCHAU veröffentlicht.

64 Seiten, 14,20 DM

FUNKSCHAU-Sonderheft **NEU**
ZX 81 Kochbuch

Dieses Sonderheft ist ein maßgeschneidertes Informationspaket für alle, die nicht nur **an** diesem Heimcomputer, sondern **mit** ihm arbeiten wollen. Es bietet 52 „Kochrezepte", 20 davon sind brandneu und zuvor noch nicht in der FUNKSCHAU veröffentlicht worden.

Für aktive Computer-Hobbyisten.

80 Seiten, 14,80 DM

ELEKTRONIK-Sonderheft
Daten-Kommunikation

Alles Wichtige, was die ELEKTRONIK bisher über dieses Spezialgebiet schrieb, finden Sie jetzt komplett in einem Heft. Damit gibt der Franzis-Verlag allen Interessierten ein vielseitiges Informationsmittel in die Hand, das alle wesentlichen Aspekte der Datenübertragung transparent macht.

2., ergänzte Auflage.
Für Einsteiger und „alte Hasen".

144 Seiten, 18 DM

Sonderhefte zum Thema Mikrocomputer

Für Einsteiger, Elektroniker und Programmierer sind diese Sonderhefte eine kompakte Informationsquelle zu einzelnen Spezialbereichen.

Bezugsmöglichkeiten

Bei allen Bahnhofsbuchhandlungen, beim Elektronik-Fachhandel, bei größeren Zeitschriftenverkaufsstellen, in Buchhandlungen oder direkt beim Franzis-Verlag gegen

- Voreinzahlung des genannten Betrages zzgl. 2,– DM Porto auf unser Postscheckkonto München Nr. 813 75-809 mit genauer Nennung des jeweiligen Titels oder
- Zusendung eines Schecks

Franzis-Verlag

Karlstraße 37,
8000 München 2,
Tel. 0 89/51 17-2 39/-3 80

In der Schweiz:
Verlag Thali AG,
CH-6285 Hitzkirch

In Österreich:
Fachbuch Center Erb,
Amerlingstraße 1,
A-1061 Wien

Analyse einer Transistorstufe

```
1 REM *** ANALYSE EINER TRANSISTORSTUFE
2 :
3 PRINTCHR$(147)"    ANALYSE EINER TRANSISTORSTUFE
4 :
5 PRINT:PRINT:PRINT"WOLLEN SIE DIE ERKLAERUNGEN LESEN?       (J=JA,N=NEIN)"
6 GETA1$:IFA1$=""THEN6
7 IFA1$="J"THEN10
8 IFA1$<>"J"ANDA1$<>"N"THENPRINT"NUR J ODER N":GOTO5
9 IFA1$="N"THENPRINT"":GOTO12
10 PRINT"":GOTO88
11 :
12 INPUT"WIDERSTAND R1:";R1
13 INPUT"WIDERSTAND R2:";R2
14 INPUT"WIDERSTAND R3:";R3
15 INPUT"WIDERSTAND R4:";R4
16 INPUT"VERSORGUNGSSPANNUNG UB:";UB
17 INPUT"STROMVERST.-FAKTOR (STATISCH) B";B
18 :
19 U5=.6:PRINT"":GOTO22:REM BASIS-EMITTER-SPANNUNG
20 PRINT: PRINT"DRUECKEN SIE FUER DIE NAECHSTE TABELLE  EINE TASTE
21 GETO$:IFO$=""THEN21:PRINT""
22 UL=(UB*R2)/(R1+R2):REM ERSATZSPANNUNGS-QUELLE
23 R0=(R1*R2)/(R1+R2):REM ERSATZWIDERSTAND
24 IB=(UL-U5)/(R0+R4*(B+1)):REM BERECHNUNG BASISSTROM
25 IC=IB*B:REM BERECHNUNG KOLLEKTOR-STROM
26 U3=R3*IC:REM BERECHNUNG SPANNUNG UEBER R3
27 U4=R4*(IC+IB):REM BERECHNUNG SPANNUNG UEBER R4
28 UC=UB-U3-U4:REM BERECHNUNG KOLLEKTOR-EMMITTER-SPANNUNG
29 U2=U4+U5:REM BERECHNUNG SPANNUNG UEBER R2
30 U1=UB-U2:REM BERECHNUNG SPANNUNG UEBER R1
31 I1=U1/R1:REM BERECHNUNG STROM DURCH R1
32 I2=U2/R2:REM BERECHNUNG STROM DURCH R2
33 I3=IC:REM STROM DURCH R3 = KOLLEKTOR-STROM
34 I4=U4/R4:REM BERECHNUNG STROM DURCH R4 = EMITTER-STROM
35 IG=I1+I3:REM BERECHNUNG GESAMTSTROM AUS VERSORGUNGSSPANNUNGSQUELLE
36 PG=UB*IG:REM BERECHNUNG DER LEISTUNG AUS VERSORGUNGSSPANNUNGSQUELLE
37 P1=I1*U1:REM BERECHNUNG DER LEISTUNG IN R1
38 P2=I2*U2:REM BERECHNUNG DER LEISTUNG IN R2
39 P3=I3*U3:REM BERECHNUNG DER LEISTUNG IN R3
40 P4=I4*U4:REM BERECHNUNG DER LEISTUNG IN R4
41 P5=IC*UC:REM BERECHNUNG DER KOLLEKTOR-EMITTER-VERLUSTLEISTUNG
42 P6=IB*U5+IC*UC:REM BERECHNUNG DER TRANSISTOR-VERLUST-LEISTUNG
43 :
44 PRINT"":PRINT"RECHENWERTE:"TAB(20)"SPANNUNGEN:":PRINT
45 PRINT"R1="TAB(7)R1 TAB(20)"U1="U1
46 PRINT"R2="TAB(7)R2 TAB(20)"U2="U2
47 PRINT"R3="TAB(7)R3 TAB(20)"U3="U3
48 PRINT"R4="TAB(7)R4 TAB(20)"U4="U4
49 PRINT"UB="TAB(7)UB TAB(20)"UCE="UC
50 PRINT"UBE="TAB(7)U5TAB(20)"UBE="U5
51 PRINT"B="  TAB(7)B
52 PRINT:PRINT
53 PRINT"STROEME:"TAB(20)"LEISTUNGEN:":PRINT
54 PRINT"I1="TAB(5)I1  TAB(20)"P1="P1
55 PRINT"I2="TAB(5)I2  TAB(20)"P2="P2
56 PRINT"I3="TAB(5)I3  TAB(20)"P3="P3
57 PRINT"I4="TAB(5)I4  TAB(20)"P4="P4
58 PRINT"IG="TAB(5)IG  TAB(20)"PG="PG
59 PRINT"IB="TAB(5)IB  TAB(20)"PT="P6
```



```
60 PRINT
61 M=A+SW
62 A=M
63 IFA>EWTHENA=0:EW=0:SW=0:GOTO66
64 IFEW<>0THEN79
65 :
66 PRINT"WOLLEN SIE NUN EINEN DER EINGABE-WERTE  AENDERN  (JA=J, NEIN=N)"
67 GETZ$:IFZ$=""THEN67
68 IFZ$<>"J"ANDZ$<>"N"THENPRINT"NUR J ODER N":GOTO66
69 IFZ$="N"THENEND
70 IFZ$="J"THENINPUT"WELCHEN (R1,R2,R3,R4,UB,B)";Y$:PRINT
71 PRINT"VARIABEL ODER FESTWERT?                  (V=VARIABEL, F=FEST)"
72 GETX$:IFX$=""THEN72
73 IFX$="V"THEN76
74 IFX$="F"THENPRINTY$"=":INPUTA:GOSUB79
75 GOTO20
76 PRINT:INPUT"ANFANGSWERT";A
77 PRINT:INPUT"ENDWERT";EW
78 PRINT:INPUT"SCHRITTWEITE";SW
79 IFY$="R1"THENR1=A
80 IFY$="R2"THENR2=A
81 IFY$="R3"THENR3=A
82 IFY$="R4"THENR4=A
83 IFY$="UB"THENUB=A
84 IFY$="B"THENB=A
85 IFEW<>0THEN20
86 RETURN
87 :
88 PRINT"  ANALYSE EINER TRANSISTORSTUFE    "
89 PRINT:PRINT:PRINT"DIESES PROGRAMM BERECHNET DEN GLEICH-
90 PRINT"SPANNUNGSARBEITSPUNKT EINER
91 PRINT"TRANSISTORSTUFE MIT GLEICHSTROM-         GEGENKOPPLUNG!":PRINT
92 PRINT"ZUR OPTIMIERUNG DER STUFE KOENNEN NACH  DER ERSTEN EINGABE WERTE
93 PRINT"VERAENDERT WERDEN, DANACH FOLGT DIE     NEUBERECHNUNG DER GESAMTEN
94 PRINT"STUFE.":PRINT:PRINT"ES KANN JEWEILS NUR EIN WERT GEAENDERT  WERDEN.
95 PRINT"DIESER KANN JEDOCH VARIABEL MIT ANGABE  VON ANFANGS- UND
96 PRINT"ENDWERT SOWIE DURCH EINGABE DER SCHRITT-WEITE GEWAEHLT WERDEN.":PRINT
97 PRINT"FERTIG?  DRUECKEN SIE EINE TASTE!
98 GETA$:IFA$=""THEN98
99 PRINT"":PRINT"SOLL EINER DER WIDERSTAENDE AUS DER      SCHALTUNG ENTFERNT
100 PRINT"WERDEN, SO IST ER ENTWEDER DURCH
101 PRINT"EINEN SEHR SEHR HOHEN WERT ODER DURCH   EINEN SEHR NIEDRIGEN WERT
102 PRINT"ZU ERSETZEN.":PRINT
103 PRINT"DIES HAENGT DAVON AB, OB DER WIDERSTAND VOELLIG AUS DER SCHALTUNG
104 PRINT"ENTFERNT WERDEN ODER ABER
105 PRINT"DURCH EINEN KURZSCHLUSS ERSETZT WERDEN  SOLL!
106 PRINT"DIE BERECHNUNG ERFOLGT NACH DER METHODE DER ERSATZSPANNUNGSQUELLE.
107 PRINT:PRINT"ALS BASIS EMITTER-SPANNUNG WURDE        0.6 VOLT GEWAEHLT.
108 PRINT"DIES KANN JEDOCH LEICHT IN ZEILE 130    ABGEAENDERT WERDEN.
109 PRINT:PRINT"DRUECKEN SIE EINE TASTE
110 GETA$:IFA$=""THEN110
111 PRINT"":PRINT"DIE IN DER AUSGABE VERWENDETEN KURZ-    ZEICHEN HABEN
112 PRINT"FOLGENDE BEDEUTUNG:":PRINT
113 PRINTTAB(10)"R1=  WIDERSTAND R1
114 PRINTTAB(10)"R2=  WIDERSTAND R2
115 PRINTTAB(10)"R3=  WIDERSTAND R3
116 PRINTTAB(10)"R4=  WIDERSTAND R4
117 PRINTTAB(10)"UB=  VERSORGUNGSSPANNUNG
118 PRINTTAB(10)"UBE= BASIS-EMITTER-SPANNUNG
```

```
119 PRINTTAB(10)"UCE= KOLLEKTOR-EMITTER-                    SPANNUNG
120 PRINTTAB(10)"U1=  SPANNUNG AN R1
121 PRINTTAB(10)"U2=  SPANNUNG AN R2
122 PRINTTAB(10)"U3=  SPANNUNG AN R3
123 PRINTTAB(10)"U4=  SPANNUNG AN R4
124 PRINT:PRINT:PRINT"DRUECKEN SIE EINE TASTE
125 GETA$:IFA$=""THEN125
126 PRINT"■
127 PRINT"FORTSETZUNG DER BEDEUTUNGEN:":PRINT
128 PRINTTAB(10)"I1=  STROM DURCH R1
129 PRINTTAB(10)"I2=  STROM DURCH R2
130 PRINTTAB(10)"I3=  STROM DURCH R3 =                      KOLLEKTORSTROM
131 PRINTTAB(10)"I4=  STROM DURCH R4 =                      EMITTERSTROM
132 PRINTTAB(10)"IB=  BASISSTROM
133 PRINTTAB(10)"P1=  LEISTUNG/R1
134 PRINTTAB(10)"P2=  LEISTUNG/R2
135 PRINTTAB(10)"P3=  LEISTUNG/R3
136 PRINTTAB(10)"P4=  LEISTUNG/R4
137 PRINTTAB(10)"PG=  LEISTUNG/GESAMT
138 PRINTTAB(10)"PT=  LEISTUNG/TRANSISTOR
139 PRINT:PRINT"DRUECKEN SIE EINE TASTE
140 GETA$:IFA$=""THEN140
141 PRINT"■
142 PRINT"DIESES PROGRAMM SOLL NUR EIN BEISPIEL   DES PET-EINSATZES ZUR
143 PRINT"BERECHNUNG ELEKTRONISCHER SCHALTUNGEN   SEIN.":PRINT
144 PRINT"IST JEDOCH FUER EINFACHE ANALYSEN SEHR  NUETZLICH.
145 PRINT:PRINT"■ACHTUNG!!!!!!!!!!!!!!!!!!!!!!!!!"
146 PRINT"■EIN NEGATIVER ZAHLENWERT BEI    "
147 PRINT"■DEN ERGEBNISSEN IST EIN:        "
148 PRINT"■NICHTMOEGLICHER ARBEITSPUNKT!!!"
149 PRINT:PRINT
150 PRINT:PRINT:PRINT"ZU IHREM BESSEREN VERSTAENDNIS NUN EIN  VEREINFACHTES
151 PRINT"SCHALTBILD:":PRINT:PRINT:PRINT
152 PRINT"DRUECKEN SIE EINE TASTE
153 GETA$:IFA$=""THEN153
154 PRINT"■"
155 PRINT"                    VEREINFACHTES SCHALT-
156 PRINT"                          BILD
157 PRINT"                    ──────────────────
158 PRINT" WERDEN DIE          |      |
159 PRINT" WIDERSTANDS-        |      |        |
160 PRINT" WERTE IN K-OHM      |      |        |
161 PRINT" UND DIE            R1     R3        |
162 PRINT" SPANNUNG IN         |      |
163 PRINT" VOLT EINGE-         └──────┘       UB
164 PRINT" GEBEN, DANN         ┌──────┐
165 PRINT" ERHALTEN SIE IN     |      | E      |
166 PRINT" DEN TABELLEN:       |      |        |
167 PRINT" K-OHM, VOLT        R2     R4        |
168 PRINT" MILLIWATT UND       |      |        ♥
169 PRINT" MILLIAMPERE.        |      |
170 PRINT"                    ──────────────────
171 PRINT"                          E= EMITTER
172 PRINT:PRINT:PRINT:PRINT"DRUECKEN SIE EINE TASTE
173 GETA$:IFA$=""THEN173
174 PRINT"■":GOTO12


READY.
```


Digitaluhr

```
1000 REM  COMPUTERSCHAU - DIGITALUHR
1010 :
1020 :
1030 REM ***************************
1040 REM STEUERZEICHEN DEFINITIONEN
1050 REM ***************************
1060 :
1070 CR$=CHR$(29):REM CURSOR NACH RECHTS
1080 CD$=CHR$(17):REM  CURSOR DOWN
1090 CH$=CHR$(19):REM  CURSOR HOME
1100 NO$=CHR$(145):REM CURSOR NACH OBEN
1110 CS$=CHR$(147):REM CLEAR SCREEN
1120 CL$=CHR$(157):REM CURSOR LINKS
1130 :
1140 REM ***************************
1150 REM GRAFIK-ZEICHEN DEFINITIONEN
1160 REM ***************************
1170 :
1180 :
1190 REM OBERE ZEICHENKETTE (WAAGERECHT)
1200 G1$="●●●●"
1210 :
1220 REM UMSTEUERUNG NACH UNTEN + LINKS
1230 G2$=CD$+CL$+CL$+CL$+CL$
1240 :
1250 REM RAHMENFARBE GRUEN
1260 POKE53280,5
1270 :
1280 REM BILDSCHIRM SCHWARZ
1290 POKE53281,0
1300 :
1310 REM ZIFFER 0
1320 G$(0)=G1$+G2$+"●  ●"+G2$+"●  ●"+G2$+"●  ●"+G2$+G1$
1330 :
1340 REM ZIFFER 1
1350 G$(1)="   ●"+G2$+"  ●●"+G2$+" ● ●"+G2$+"   ●"+G2$+"   ●"
1360 :
1370 REM ZIFFER 2
1380 G$(2)=G1$+G2$+"   ●"+G2$+G1$+G2$+"●   "+G2$+G1$
1390 :
1400 REM ZIFFER 3
1410 G$(3)=G1$+G2$+"   ●"+G2$+" ●●●"+G2$+"   ●"+G2$+G1$
1420 :
1430 REM ZIFFER 4
1440 G$(4)="●  ●"+G2$+"●  ●"+G2$+G1$+G2$+"   ●"+G2$+"   ●"
1450 :
1460 REM ZIFFER 5
1470 G$(5)=G1$+G2$+"●   "+G2$+G1$+G2$+"   ●"+G2$+G1$
1480 :
1490 REM ZIFFER 6
1500 G$(6)=G1$+G2$+"●   "+G2$+G1$+G2$+"●  ●"+G2$+G1$
1510 :
1520 REM ZIFFER 7
1530 G$(7)=G1$+G2$+"   ●"+G2$+"   ●"+G2$+"   ●"+G2$+"   ●"
1540 :
1550 REM ZIFFER 8
1560 G$(8)=G1$+G2$+"●  ●"+G2$+G1$+G2$+"●  ●"+G2$+G1$
1570 :
1580 REM ZIFFER 9
1590 G$(9)=G1$+G2$+"●  ●"+G2$+G1$+G2$+"   ●"+G2$+G1$
1600 :
1610 REM DOPPELPUNKT DEFINIEREN
```

```
1620 DP$=CD$+"◆"+CD$+CD$+CL$+"◆"+NO$+NO$+NO$+CR$:PRINTCS$
1630 :
1640 REM **********************
1650 REM ENDE DER DEFINITIONEN
1660 REM **********************
1670 :
1680 :
1690 :
1700 REM SCHIRM LOESCHEN + CURSOR TIEFER
1710 PRINTCSCD$CD$CD$
1720 :
1730 PRINT"      COMPUTERSCHAU   DIGITALUHR
1740 PRINTCD$CD$
1750 PRINT"BITTE DIE UHRZEIT EINGEBEN     000000";
1760 :
1770 REM CURSOR LINKS SETZEN AUF 1. NULL
1780 FORI=1TO8
1790 PRINTCL$;
1800 NEXT
1810 :
1820 REM UHRZEIT EINGEBEN
1830 INPUTT1$
1840 :
1850 REM ZEICHENLAENGE PRUEFEN
1860 IFLENT1$)<>6THEN1710
1870 :
1880 REM ZEIT UEBERGEBEN
1890 TI$=T1$
1900 :
1910 PRINTCS$CD$"          STOP MIT SPACE-TASTE
1920 PRINTCD$CD$CD$CD$;
1930 PRINT"   STUNDEN     MINUTEN     SEKUNDEN"
1940 :
1950 REM CURSOR HOME + NACH UNTEN
1960 PRINTCH$
1970 FORJ=1TO5
1980 PRINTCD$
1990 NEXT
2000 :
2010 REM CURSOR RECHTS SETZEN
2020 PRINTCR$CR$CR$;
2030 :
2040 REM ** UHRZEIT AUSGEBEN **
2050 FORT=1TO6
2060 PRINTG$(VAL(MID$(TI$,T,1)))" "NO$NO$NO$NO$;
2070 :
2080 REM DOPPELPUNKT AUSGEBEN
2090 IFT=2ORT=4THENPRINTDP$;
2100 NEXT
2110 :
2120 REM ** ENDE DER ZEITAUSGABE**
2130 :
2140 REM ABFRAGE OB ENDE
2150 GETX$
2160 IFX$=" "THENPRINTCS$:END
2170 :
2180 REM WARTESCHLEIFE
2190 T$=TI$
2200 IFTI$=T$THEN2200
2210 :
2220 REM WEITERMACHEN
2230 GOTO1960

READY.
```


Computergedicht

```
100 PRINT"■COMPUTERGEDICHT■"
110 DIMV$(10),V3$(10),AD$(10),A$(10),S$(10),SR$(10,2),AR$(10,2)
120 FORI=1TO10:READV$(I):NEXTI
130 FORI=1TO10:READV3$(I):NEXTI
140 FORI=1TO10:READAD$(I):NEXTI
150 FORI=1TO10:READA$(I):NEXTI
160 FORI=1TO10:READS$(I):NEXTI
170 FORI=1TO7
180 READSR$(I,1),SR$(I,2):NEXTI
190 FORI=1TO6
200 READAR$(I,1),AR$(I,2):NEXTI
210 FORI=1TO7:READT$(I):NEXTI
220 FORI=1TO3
230 R0(I)=INT(10*RND(1)+1)
240 R1(I)=INT(10*RND(1)+1)
250 R2(I)=INT(7*RND(1)+1)
260 R3(I)=INT(10*RND(1)+1)
270 R4(I)=INT(10*RND(1)+1)
280 R5(I)=INT(10*RND(1)+1)
290 R6(I)=INT(10*RND(1)+1)
300 R7(I)=INT(6*RND(1)+1)
310 R8(I)=INT(10*RND(1)+1)
320 R9(I)=INT(10*RND(1)+1)
330 NEXTI
340 REM GEDICHT
350 FORI=1TO3
360 PRINTT$(1);" ";V$(R0(I));" ";T$(2);" ";AD$(R1(I));" ";SR$(R2(I),1);"?"
370 PRINTT$(3);" ";V$(R3(I));" ";T$(4);" ";V$(R5(I));" ";AR$(R7(I),1);"."
380 PRINTT$(5);" ";A$(R5(I));" ";V3$(R6(I));" ";SR$(R2(I),2);","
390 PRINTT$(6);" ";T$(7);" ";S$(R8(I));" ";V$(R9(I));" ";AR$(R7(I),2);"!"
400 PRINT
410 NEXTI
420 DATAABRODELN,WEHEN,FUEHLEN,WEINEN,LACHEN,LIEBEN,SAEUSELN,WIMMERN,ZITTERN
430 DATABEBEN
440 DATAHASST,FRAGT,LIEBT,WEINT,SUCHT,FINDET,GEHT,KAUT,MALT,RETTET
450 DATAROTE,BLAUEN,FERNER,HOHEN,EDLEN,REINEN,TREUEN,HOLDEN,FREIEN,LAUEN
460 DATASUESS,SELIG,EDEL,REIN,SANFT,HEITER,STOLZ,EWIG,HOLD,EIFRIG
470 DATAWINDE,STERNE,TAGE,ENGEL,MONDE,QUALEN,WOLKEN,WELTEN,WOGEN,HERZEN
480 DATAKLAGEN,DER MAGEN,VOEGEL,DAS SEGEL,WINDE,DIE LINDE,UFER,DER RUFER
490 DATATAGE,DIE PLAGE
500 DATAROSEN,DAS KOSEN,SCHRITTE,DIE MITTE
510 DATAGRAU,LAU,SELIG,FROEHLICH,WEIT,BREIT,HEITER,WEITER,TREU,SCHEU,LANG,BANG
520 DATAWARUM,DIE,SIE,UND,ABER,DENN,DIE

READY.
```

Suchprogramm
Headerfehler

```
100 PRINT"■COMPUTERSCHAU":POKE53281,1
110 PRINT"SUCHPROGRAMM: HEADERFEHLER
120 PRINT"STOP MIT 'E'
130 PRINT"■SPUR   : ":PRINT"■SEKTOR :"
140 PRINT"■FEHLER :  OK"
150 OPEN15,8,15,"I":OPEN2,8,2,"#"
160 FORT=1TO35
170 IFT<=17THENY=20
180 IFT>17THENY=18
190 IFT>24THENY=17
200 IFT>30THENY=16
210 FORS=0TOY
220 F$="U1 2 0"+STR$(T)+STR$(S)
230 PRINT#15,F$:INPUT#15,F1$,F2$
240 GETA$:IFA$="E"THEN310
250 T$=MID$(STR$(T),2):S$=MID$(STR$(S),2)
260 IFLEN(T$)<2THENT$="0"+T$
270 IFLEN(S$)<2THENS$="0"+S$
280 PRINT"■■■■■■"TAB(10)T$:PRINT"■"TAB(10)S$
290 IFVAL(F1$)<>0THENPRINT"■"TAB(10)F1$,F2$:GOSUB330
300 NEXTS:NEXTT
310 PRINT:PRINT:CLOSE2:CLOSE15:END
320 :
330 PRINT"■WEITER MIT SPACE"
340 GETA$:IFA$=""THEN340
350 PRINT"■■■■■■■■■■■■■■OK                    ":PRINT"
360 PRINT"                                    ":RETURN
READY.
```

Zahlenwandlung
Hexadezimal
in Dezimal

```
100 DIMA$(15),V(4)
110 FORI=0TO15
120 READA$(I):NEXT
130 PRINT"■COMPUTERSCHAU"
140 PRINT"ZAHLENWANDLUNG■"
150 PRINT"■SEDEZIMAL (HEXADEZIMAL) IN DEZIMAL■■"
160 INPUT"SEDEZIMALZAHL (MAX. 4 STELLEN )  0000 ■■■■■■■■";HD$
170 IFLEN(HD$)>4THENPRINT"■■":GOTO130
180 IFLEN(HD$)<1THEN130
181 IFLEN(HD$)<4THENHD$="0"+HD$:GOTO181
190 FORI=1TO4:FL=0
200 V$=MID$(HD$,I,1)
210 FORX=0TO15
220 IFV$=A$(X)THENV(I)=X:FL=1
230 NEXTX:IFFL=0THENPRINT"FEHLER":GOTO270
240 NEXTI:PRINT
250 DZ=V(4)+V(3)*16+V(2)*256+V(1)*4096
260 PRINT"$"HD$"   =DEZIMAL:"DZ
270 PRINT"■■WEITER = <RETURN>
271 PRINT"■E■NDE■■■
280 GETX$:IFX$=""THEN280
290 IFX$="E"THENEND
300 FORI=1TO4:V(I)=0:NEXT
310 IFX$=CHR$(13)THEN130
320 GOTO280
330 DATA0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
READY.
```


Zahlenwandlung
Dezimal – Hexadezimal

```
100 DIMA$(15),H(4)
110 FORI=0TO15
120 READA$(I):NEXT
130 PRINT"■COMPUTERSCHAU"
140 PRINT"ZAHLENWANDLUNG■"
150 PRINT"■DEZIMAL IN SEDEZIMAL (HEXADEZIMAL) ■■"
160 DE$="":INPUT"DEZIMALZAHL    (MAX. 65535)       0 ■■■■■";DE$
170 DE=VAL(DE$):IFDE>65535THEN290
180 IFLEN(DE$)>5THENPRINT"■■":GOTO290
190 IFLEN(DE$)<1THEN290
200 P$=RIGHT$(STR$(DE),LEN(DE$)):IFDE$<>P$THEN290
210 IFDE<0THEN290
220 H(1)=INT(DE/4096)
230 H(2)=INT((DE-H(1)*4096)/256)
240 H(3)=INT((DE-H(1)*4096-H(2)*256)/16)
250 H(4)=DE-H(1)*4096-H(2)*256-H(3)*16
260 PRINT:PRINTDE" = $";:FORI=1TO4
270 PRINTA$(H(I));
280 NEXTI:PRINT:GOTO300
290 PRINT"FEHLER"
300 PRINT"■■WEITER = <RETURN>
310 PRINT"■E■NDE■■■
320 GETX$:IFX$=""THEN320
330 IFX$="E"THENEND
340 FORI=1TO4:H(I)=0:NEXT
350 IFX$=CHR$(13)THEN130
360 GOTO320
370 DATA0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

READY.
```

Startadressen von
Diskettenprogrammen

```
90 PRINT"■COMPUTERSCHAU
91 PRINT"STARTADRESSEN VON DISKETTENPROGRAMMEN■"
92 PRINT"DER PRG-NAME DARF MIT '*' ABGEKUERZT    WERDEN■"
100 INPUT"PRG-NAME";PN$
110 OPEN1,8,0,PN$
120 FORI=1TO2
130 GET#1,A$(I):IFA$(I)=""THENA$(I)=CHR$(0)
140 NEXTI
150 PRINT"BEGINN:  "ASC(A$(1))+256*ASC(A$(2))
170 CLOSE1

READY.
```

Mail-Box-Datei

```
100 PRINT"■"
110 PRINT"              ■****************"
120 PRINT"              ■****************"
130 PRINT"              ■*MAIL-BOX-DATEI*■"
140 PRINT"              ■****************"
150 PRINT"              ■****************"
160 PRINT
170 PRINT:PRINT:PRINT"  MINDESTENS ZWEI BUCHSTABEN EINGEBEN
180 PRINT:PRINT
190 INPUT"  ■MAILBOX-NAME";MN$
200 S$=LEFT$(MN$,2)
210 PRINT
220 READM$
230 IFM$="END"THEN270
240 IFS$=LEFT$(M$,2)THENPRINTM$
250 GOTO220
260 END
270 PRINT"■  FERTIG  <BITTE TASTE DRUECKEN>"
280 GETA$
290 IFA$=""THEN280
300 RUN
310 DATA CODA.............089/596465
320 DATA TEDAS............089/596422
330 DATA TEDAS............089/598423
340 DATA EPSON............089/593453
350 DATA SOFTWARE EXPRESS..0211/414579
360 DATA SATURN ..........0221/1616284
370 DATA CC...............02202/50033
380 DATA SYMIC............02161/200928
390 DATA END

READY.
```


ID-Prüfer

```
1000 PRINTCHR$(147)"COMPUTERSCHAU      ++ ID-PRUEFER ++"
1010 PRINT
1020 PRINT"MIT DIESEM PROGRAMM WIRD UEBERPRUEFT
1030 :
1040 PRINT"OB EINE DISKETTEN-ID MEHRFACH VOR-
1050 PRINT"HANDEN IST.
1060 PRINT
1070 PRINT"FALLS DIES ZUTRIFFT, KANN SIE
1080 PRINT"AUF WUNSCH GEAENDERT WERDEN.
1090 PRINT
1100 PRINT"SINN UND ZWECK:
1110 PRINT"DAMIT BEI DER DISKETTENARCHIVIERUNG
1120 PRINT"KEIN UEBERSCHREIBEN VON BEREITS AUF-
1130 PRINT"GENOMMENEN DISK-INHALTEN DURCH INHALTE
1140 PRINT"VON ANDEREN DISKETTEN MIT GLEICHEN ID'S ERFOLGT.
1150 :
1160 PRINT"ES SOLL NUR EIN GRUNDPROGRAMM SEIN.
1170 PRINT"JEDER BENUTZER SOLLTE ES FUER SICH
1180 PRINT"SELBST NOCH BESSER AUSBAUEN
1190 PRINT"UND VOR ALLEM NOCH VERBESSERN!
1200 PRINT"ODER IN DAS ARCHIVIERUNGSPROGRAMM
1210 PRINT"EINBAUEN
1220 PRINT
1230 :
1240 INPUT"MINIMALANZAHL DER DISKETTEN";MA
1250 :
1260 REM ANZAHL DER DISK-ID'S FESTLEGEN
1270 DIMID$(MA)
1280 :
1290 REM DOKUMENTATION
1300 I=I+1
1310 :
1320 REM SCHIRM FREI UND NEUE DISK ANFORDERN
1330 PRINTCHR$(147)"BITTE DISKETTE NR.: "I" EINLEGEN UND
1340 PRINT"RETURN DRUECKEN           (E = ENDE)
1350 :
1360 REM AUF TASTENDRUCK WARTEN
1370 A$="":GETA$:IFA$=""THEN1370
1380 :
1390 REM FALLS ENDE DANN SPRUNG
1400 IFA$="E"THEN2160
1410 :
1420 REM KEIN RETURN  +ZURUECK+
1430 IFA$<>CHR$(13)THEN1370
1440 :
1450 REM KOMMANDOKANAL OEFFEN
1460 REM UND INITIALISIEREN
1470 OPEN15,8,15,"I0"
1480 :
1490 REM PUFFER OEFFNEN
1500 OPEN2,8,2,"#"
1510 :
1520 REM LESEZUGRIFF UEBER PUFFER 2
1530 PRINT#15,"U1 2 0 18 0"
1540 :
1550 REM PUFFERZEIGER SETZEN
1560 PRINT#15,"B-P 2 162"
1570 :
1580 REM ID HOLEN
1590 GET#2,Q1$,Q2$:CLOSE2:CLOSE15
1600 :
1610 REM ID AUSGEBEN
1620 PRINTCHR$(17)CHR$(17)CHR$(17)"ID= "Q1$Q2$
```

```
1630 REM I$=GEHOLTE ID
1640 I$=Q1$+Q2$
1650 :
1660 REM CHECK OB SCHON VORHANDEN
1670 FORJ=1TOI
1680 IFI$=ID$(J)THENPRINT:PRINT:PRINTCHR$(18)"SCHON VORHANDEN":GOTO1750
1690 NEXTJ
1700 :
1710 REM ID IST NEU
1720 ID$(I)=I$:GOTO1300
1730 :
1740 REM ID SCHON VORHANDEN
1750 ID$(I)=I$
1760 :
1770 PRINT"SOLL DIE ID GEAENDERT WERDEN "CHR$(18)"J/N"
1780 :
1790 REM SOLL AENDERUNG ERFOLGEN
1800 A$="":GETA$:IFA$=""THEN1800
1810 IFA$="J"THEN1840:REM AENDERN
1820 IFA$<>"J"THEN1300:REM NICHT AENDERN
1830 :
1840 INPUT"NEUE ID";ID$
1850 :
1860 REM NUR DIE ERSTEN BEIDEN ZEICHEN
1870 REM WERDEN ANGENOMMEN
1880 NI$=LEFT$(ID$,2)
1890 :
1900 REM WIEDER INITIALISIEREN
1910 OPEN15,8,15,"I0"
1920 :
1930 REM WIEDER PUFFER OEFFNEN
1940 OPEN2,8,2,"#"
1950 :
1960 REM BLOCKPOINTER SETZEN
1970 PRINT#15,"B-P:2,162"
1980 :
1990 REM NEUE ID IN PUFFER SCHREIBEN
2000 PRINT#2,NI$;
2010 :
2020 REM AUF DISK SCHREIBEN
2030 PRINT#15,"U2 2 0 18 0"
2040 :
2050 REM FILES SCHLIESSEN
2060 CLOSE2:CLOSE15
2070 :
2080 REM NEUE ID AUFNEHMEN IN TABELLE
2090 ID$(I)=NI$
2100 :
2110 REM WEITERMACHEN
2120 GOTO1300
2130 :
2140 :
2150 REM AUSGABE
2160 PRINTCHR$(147)"ES FOLGT DIE AUSGABE ALLER ID'S
2170 :
2180 FORJ=1TOI-1
2190 PRINT"TASTE DRUECKEN   "CHR$(146);
2200 :
2210 REM WARTEN AUF TASTENDRUCK
2220 X$="":GETX$:IFX$=""THEN2220
2230 :
2240 REM JEWEILS EINE ID AUSGEBEN
2250 PRINTID$(J):NEXTJ
2260 PRINT"DAS WAR'S
```



```
1000 PRINTCHR$(147)"-OMPUTER♥CHAU          FILE-RETTER                File-Retter
1010 :
1020 REM UMSCHALTUNG AUF KLEINSCHRIFT
1030 PRINTCHR$(14)
1040 :
1050 REM DIMENSIONIERUNG
1060 DIMZ(300),FT$(4)
1070 :
1080 REM AZ  = ANFUEHRUNGSZEICHEN
1090 AZ$=CHR$(34)
1100 :
1110 REM REVERS ON
1120 RV$=CHR$(18)
1130 :
1140 REM REVERS OFF
1150 VR$=CHR$(146)
1160 :
1170 REM CURSOR DOWN
1180 CD$=CHR$(17)
1190 :
1200 REM CURSOR NACH OBEN
1210 NO$=CHR$(145)
1220 :
1230 REM FT$ = FILETYPEN
1240 FT$(0)="DEL"
1250 FT$(1)="SEQ"
1260 FT$(2)="PRG"
1270 FT$(3)="USR"
1280 FT$(4)="REL"
1290 :
1300 REM EF= ENDE-FLAG AUF NULL SETZEN
1310 EF=0
1320 :
1330 REM KDO-KANAL OEFFNEN + INITIAL.
1340 OPEN15,8,15,"I0"
1350 :
1360 REM PUFFER OEFFNEN
1370 OPEN2,8,2,"#2"
1380 FI=0
1390 :
1400 REM DEFINITION SPUR/SEKTOR  18 01
1410 TR=18:SE=1
1420 :
1430 REM LESEZUGRIFF  T+S SETZEN
1440 PRINT#15,"U1:2 0"TR;SE
1450 :
1460 REM FEHLERKANAL ABFRAGEN
1470 GOSUB2820
1480 :
1490 FORI=0TO1
1500 GET#2,A$
1510 :
1520 GOSUB2770
1530 P(I)=ASC(A$)
1540 NEXT
1550 A=2
1560 GOSUB2800
1570 :
1580 GET#2,A$
1590 GOSUB2770
1600 A=ASC(A$)AND7
1610 :
1620 PN$=""
```

```
1630 FORI=0TO1
1640 GET#2,A$
1650 GOSUB2770
1660 L(I)=ASC(A$)
1670 NEXT
1680 :
1690 FORI=3TO18
1700 GET#2,A$
1710 PN$=PN$+A$
1720 NEXT
1730 IFPN$=""THEN1950
1740 :
1750 REM TRACK-STRING 2-STELLIG
1760 TR$=STR$(L(0))
1770 TR$=MID$(TR$,2)
1780 IFLEN(TR$)<2THENTR$="0"+TR$
1790 :
1800 REM SEKTOR-STRING 2-STELLIG
1810 SE$=STR$(L(1))
1820 SE$=MID$(SE$,2)
1830 IFLEN(SE$)<2THENSE$="0"+SE$
1840 :
1850 :
1860 REM BILDSCHIRMAUSGABE
1870 PRINTTAB(2)TR$TAB(5)SE$;
1880 PRINTTAB(8)"FILETYP:";
1890 IFFT$(A)="DEL"THENPRINTRV$FT$(A)VR$AZ$PN$;
1900 IFFT$(A)<>"DEL"THENPRINTFT$(A)RV$AZ$PN$;
1910 PRINTTAB(28)AZ$VR$
1920 :
1930 IFA=0THENGOTO2490
1940 :
1950 FI=FI+1
1960 IFFI<8THENA=FI*32+2:GOSUB2800:GOTO1580
1970 :
1980 IFP(0)=0THEN2050
1990 :
2000 PRINTCD$"  "RV$"SPURWECHSEL"VR$;
2010 TR=P(0)
2020 SE=P(1):FI=0
2030 PRINTTR;SE
2040 GOTO1440
2050 CLOSE8
2060 PRINT"  "RV$"DIRECTORY-ENDE"VR$
2070 :
2080 REM FALLS ENDE DANN VERIFY (COLLECT)
2090 IFEFTHENPRINT"  VERIFY ":PRINT#15,"V0"
2100 :
2110 REM FEHLER ABFRAGEN
2120 GOSUB2820
2130 :
2140 CLOSE15
2150 PRINT"■ENDE":END
2160 :
2170 PRINT#15,"U1:2 0"L(0);L(1)
2180 RETURN
2190 :
2200 D=2:PRINT"  SP/SE    :";
2210 GOSUB2170
2220 :
2230 FORI=0TO1
2240 GET#2,A$:GOSUB2770
2250 Z(I)=ASC(A$)
```



```
2260 :
2270 REM 2 STELLEN
2280 Z$=STR$(Z(I))
2290 Z$=MID$(Z$,2)
2300 IFLEN(Z$)<2THENZ$="0"+Z$
2310 :
2320 :
2330 PRINTZ$"  ";
2340 NEXT
2350 PRINT:PRINTNO$"  "RV$"  SP/SE  :"VR$;
2360 IFZ(0)=0THENPRINT:RETURN
2370 PRINT#15,"B-A:0";Z(0);Z(1)
2380 L(0)=Z(0):L(1)=Z(1):Z(D)=Z(0)
2390 Z(D+1)=Z(1)
2400 D=D+2
2410 IFDS=0THEN2210
2420 PRINTCD$CD$RV$"FILE BEREITS ZERSTOERT !! "VR$
2430 X=NOT(X)
2440 A=FI*32+2
2450 FORI=2TOD-4STEP2
2460 PRINT#15,"B-F:0";Z(I);Z(I+1)
2470 NEXT
2480 L(0)=TR:L(1)=SE:GOSUB2170:GOTO2800
2490 PRINT:INPUT"  RETTEN      (J/N)   ";JA$
2500 IFJA$<>"J"THEN1950
2510 X=0
2520 PRINT"  BLOECKE RESTAURIEREN!"
2530 Z(0)=L(0)
2540 GOSUB2200
2550 :
2560 IFXTHEN1950
2570 PRINTNO$TAB(2)RV$"----------------"
2580 PRINT"  PROG-, SEQ-,USR-, REL-FILE ?";
2590 INPUTJA$
2600 IFJA$<>"P"ANDJA$<>"S"ANDJA$<>"U"ANDJA$<>"R"THEN2590
2610 IFJA$="P"THENKI=130:PRINT"  PRG-FILE ";:GOTO2650
2620 IFJA$="U"THENKI=131:PRINT"  USR-FILE ";:GOTO2650
2630 IFJA$="R"THENKI=132:PRINT"  REL-FILE ";:GOTO2650
2640 KI=129:PRINT"  SEQ-FILE ";
2650 PRINT"RETTEN"
2660 L(0)=TR:L(1)=SE
2670 GOSUB2170
2680 A=FI*32+2
2690 GOSUB2800
2700 PRINT#2,CHR$(KI);
2710 BA=((D-2)/2)+1
2720 :
2730 PRINT#15,"U2:2 0"TR;SE
2740 PRINT" "BA"  BLOECK(E) RESTAURIERT  "
2750 EF=1
2760 GOTO1950
2770 IFA$=""THENA$=CHR$(0)
2780 RETURN
2790 :
2800 PRINT#15,"B-P 2,";A
2810 :
2820 INPUT#15,F1,F2$,F3,F4
2830 IFF1=0THENRETURN
2840 :
2850 PRINT"DISK FEHLER
2860 PRINT"FEHLER MELDUNG: "F1" "F2$","F3","F4
2870 PRINT"  ABBRUCH WEGEN FEHLER
2880 END
READY.
```

Programm zur
CBM-ASCII
und Tokenausgabe

```
100 REM***
110 :
120 REM ************************
130 REM *  !!!!! ACHTUNG  !!!!!  *
140 REM *ZEILE 100 MUSS UNBEDINGT*
150 REM *    VORHANDEN SEIN      *
160 REM *  !!!!! ACHTUNG  !!!!!  *
170 REM ************************
180 :
190 REM DEFINITIONEN + FESTLEGUNGEN
200 :
210 :
220 REM CLEAR SCREEN (BILDSCHIRM LOESCHEN)
230 CS$=CHR$(147)
240 :
250 REM CURSOR HOME
260 CH$=CHR$(19)
270 :
280 REM CURSOR DOWN
290 CD$=CHR$(17)
300 :
310 REM RAHMEN SCHWARZ
320 POKE53280,0
330 :
340 REM SCHIRM SCHWARZ
350 POKE53281,0
360 :
370 REM ZEICHEN WEISS
380 PRINTCHR$(5)
390 :
400 REM ENDE DEF UND FESTLEGUNGEN
410 :
420 PRINTCS$"COMPUTERSCHAU
430 PRINTCD$"PROGRAMM ZUR CBM-ASCII
440 PRINT"UND TOKENAUSGABE
450 :
460 INPUT"CODEZAHL";CZ
470 IFCZ<330RCZ>256THEN420
480 :
490 REM IN ZEILE 100 HINTER REM* POKEN
500 POKE2055,CZ
510 :
520 REM CURSOR IN DIE LINKE OBERE ECKE
530 PRINTCH$
540 :
550 REM CURSUR NACH UNTEN...
560 FORI=1TO5:PRINTCHR$(17):NEXTI
570 :
580 REM AUF DEN BILDSCHIRM SCHREIBEN
590 PRINT"LIST 100"
600 :
610 PRINTCD$CD$CD$
620 :
630 PRINT"RUN"
640 :
650 :
660 REM CURSOR IN LINKE OBERE BILDSCHIRMECKE
670 PRINTCH$
680 :
690 FORI=1TO8
700 PRINTCD$;
710 NEXT

READY.
```


Farbenspielerei

```
100 REM ****************************************
110 REM ***          FARBENSPIELEREI           ***
120 REM ***--------------------------------------***
130 REM ***             ZUM ABBRUCH            ***
140 REM ***              SPACE-TASTE           ***
150 REM ***                DRUECKEN            ***
160 REM ****************************************
170 :
180 REM BILDSCHIRM LOESCHEN
190 PRINTCHR$(147)
200 :
210 REM ANZAHL DER ZEILEN
220 AZ=15
230 :
240 REM BILDSCHIRMADRESSE
250 BB=1064
260 :
270 REM BILDSCHIRM WEISS
280 POKE53281,1
290 :
300 RESTORE
310 :
320 REM DATASTATEMENT HOLEN
330 READA$
340 :
350 L=LEN(A$)
360 :
370 FORI=1TOL
380 GOSUB590:REM RANDOM-ZAHL HOLEN
390 PRINTMID$(A$,I,1);:REM JEWEILS NUR EIN ZEICHEN AUSGEBEN
400 NEXTI
410 :
420 GETTA$:IFTA$=" "THENEND:REM SPACE-TASTE FUER ABBRUCH BETAETIGEN
430 :
440 FL=1:REM FL IST FLAG UM DEN PRINT-BEFEHL AUSZUGEBEN
450 IFFLTHENPRINT:FL=0:REM ZEILENVORSCHUB (PRINT)
460 :
470 FORI=LTO0STEP-1:REM SCHLEIFE - ZEICHEN RUECKWAERTS HOLEN
480 Y=PEEK(BB+I+40):REM ZEICHEN AUS VORIGER ZEILE HOLEN
490 POKE(BB+I+80),Y:REM ZEICHEN IN NEUE ZEILE POKEN
500 NEXTI
510 :
520 BB=BB+40
530 IFBB>1064+6*40THENGOSUB620:RUN:REM NEUSTART
540 GOTO300
550 POKE646,X:REM FARBE UNTER DEM CURSOR AENDERN
560 :
570 DATA"-ER -OMMODORE -64 IST WEIT VERBREITET!"
580 PRINTCHR$(147):BB=1064
590 X=INT(RND(0)*16+1)
600 POKE646,X
610 RETURN
620 REM * * * * * * * * * * * * * * * * * * * * * * * * *
630 PRINTCHR$(147):BB=1064:REM SCHIRM LOESCHEN
640 :
650 FORJ=1TO14:REM FARBEN 1 BIS 14 DURCHLAUFEN
660 POKE53281,J:REM FARBE DES BILDSCHIRMS AENDERN
670 :
680 FORU=1TO10:NEXTU:REM ZEITSCHLEIFE FUER FARBWECHSEL
690 POKE53280,J+2:REM RAHMENFARBE AENDERN
700 :
710 FORU=1TO10:NEXTU:REM ZEITSCHLEIFE
720 NEXTJ
730 RETURN
```

```
63000 REM COMPUTERSCHAU CBM-ASCII-CODE-AUSGABE C 64
63001 :
63002 PRINTCHR$(147)"CBM-ASCII-CODE C 64
63003 FORI=1TO10:PRINTCHR$(17):NEXTI
63004 PRINT"ENDE DURCH SPACE-TASTE"
63005 PRINT"COMPUTERSCHAU"
63006 X=36:DIMX(X),X$(X)
63007 FORI=1TOX:READX(I),X$(I):NEXT
63008 BL$="                                        "
63009 :
63010 PRINTCHR$(19)CHR$(17)CHR$(17)"BITTE ZEICHEN EINGEBEN     ":PRINT
63011 POKE631,34:POKE198,1
63012 GETSZ$:IFSZ$=""THEN63012
63013 IFSZ$=" "THENPRINTCHR$(147)"LETZTE TASTE WAR: CHR$(32)":END
63014 IFSZ$=CHR$(34)THEN63012
63015 FORI=1TOX
63016 IFASC(SZ$)=X(I)THENPRINT:PRINT"CHR$("X(I)")   = "X$(I):GOTO63019
63017 NEXTI
63018 PRINT:PRINTSZ$"   = CHR$("ASC(SZ$)")
63019 FORI=1TO500:NEXTI
63020 PRINTCHR$(19):FORI=1TO2:PRINTCHR$(17):NEXTI:PRINTBL$
63021 GOTO63010
63022 :
63023 DATA147,CLS BILDSCHIRM LOESCHEN
63024 :
63025 DATA029,CURSOR RECHTS
63026 DATA157,CURSOR LINKS
63027 DATA145,CURSOR UP
63028 DATA017,CURSOR DOWN
63029 DATA019,CURSOR HOME
63030 :
63031 DATA020,DELETE
63032 DATA148,INSERT
63033 :
63034 DATA144,SCHWARZ     (1)
63035 DATA005,WEISS       (2)
63036 DATA028,ROT         (3)
63037 DATA159,CYAN        (4)
63038 DATA156,PURPUR      (5)
63039 DATA030,GRUEN       (6)
63040 DATA031,BLAU        (7)
63041 DATA158,GELB        (8)
63042 :
63043 DATA129,ORANGE      (C 1)
63044 DATA149,BRAUN       (C 2)
63045 DATA150,HELLROT     (C 3)
63046 DATA151,DUNKELGRAU  (C 4)
63047 DATA152,MITTELGRAU  (C 5)
63048 DATA153,HELLGRUEN   (C 6)
63049 DATA154,HELLBLAU    (C 7)
63050 DATA155,HELLGRAU    (C 8)
63051 :
63052 DATA018,REVERS ON
63053 DATA146,REVERS OFF
63054 :
63055 DATA006,CONTROL + PFEIL NACH LINKS
63056 DATA131,RUN 0. COM-RUN/STOP
63057 :
63058 DATA133,F 1 - TASTE
63059 DATA134,F 3 - TASTE
63060 DATA135,F 5 - TASTE
63061 DATA136,F 7 - TASTE
63062 DATA137,F 2 - TASTE
63063 DATA138,F 4 - TASTE
63064 DATA139,F 6 - TASTE
63065 DATA140,F 8 - TASTE

READY.
```


Liste der Basicbefehle

Um Programme mittels Monitor im Speicher des C 64 oder auch auf den Disketten besser analysieren zu können, benötigt man außer dem Grundwissen, wie die Ablage erfolgt, auch noch weitere Informationen, wie z. B. auch die Codes für die Tokens, also für die in „Ein-Byte-Form" abgespeicherten Basicbefehle. Je nach Utilitie-Programm benötigt man diese Zahlen entweder in Dezimal oder im Hexcode. Wir haben Ihnen anschließend zwei Übersichten abgedruckt. Bei der einen sind diese Tokens dem Alphabet nach und bei der anderen nach ihren dezimalen (entspricht auch gleichzeitig den hexadezimalen) Werten sortiert aufgelistet.

```
*************************************************
*            LISTE DER BASIC-BEFEHLE            *
*       NACH DEN DEZIMALEN WERTEN SORTIERT      *
*************************************************
```

BASICBEFEHL	DEZ.-CODE	HEXCODE	BASICBEFEHL	DEZ.-CODE	HEXCODE
END	128	80	SPC(	166	A6
FOR	129	81	THEN	167	A7
NEXT	130	82	NOT	168	A8
DATA	131	83	STEP	169	A9
INPUT#	132	84	+	170	AA
INPUT	133	85	-	171	AB
DIM	134	86	*	172	AC
READ	135	87	/	173	AD
LET	136	88	↑	174	AE
GOTO	137	89	AND	175	AF
RUN	138	8A	OR	176	B0
IF	139	8B	>	177	B1
RESTORE	140	8C	=	178	B2
GOSUB	141	8D	<	179	B3
RETURN	142	8E	SGN	180	B4
REM	143	8F	INT	181	B5
STOP	144	90	ABS	182	B6
ON	145	91	USR	183	B7
WAIT	146	92	FRE	184	B8
LOAD	147	93	POS	185	B9
SAVE	148	94	SQR	186	BA
VERIFY	149	95	RND	187	BB
DEF	150	96	LOG	188	BC
POKE	151	97	EXP	189	BD
PRINT#	152	98	COS	190	BE
PRINT	153	99	SIN	191	BF
CONT	154	9A	TAN	192	C0
LIST	155	9B	ATN	193	C1
CLR	156	9C	PEEK	194	C2
CMD	157	9D	LEN	195	C3
SYS	158	9E	STR$	196	C4
OPEN	159	9F	VAL	197	C5
CLOSE	160	A0	ASC	198	C6
GET	161	A1	CHR$	199	C7
NEW	162	A2	LEFT$	200	C8
TAB	163	A3	RIGHT$	201	C9
TO	164	A4	MID$	202	CA
FN	165	A5	GO	203	CB

```
*****************************************
*         LISTE DER BASICBEFEHLE        *
*   MIT ZUGEHOERIGEN DEZ- UND HEXCODES  *
*             (ASCII-SORTIERT)          *
*****************************************
```

BASICBEFEHL	DEZ.-CODE	HEXCODE
*	172	AC
+	170	AA
-	171	AB
/	173	AD
<	179	B3
=	178	B2
>	177	B1
ABS	182	B6
AND	175	AF
ASC	198	C6
ATN	193	C1
CHR$	199	C7
CLOSE	160	A0
CLR	156	9C
CMD	157	9D
CONT	154	9A
COS	190	BE
DATA	131	83
DEF	150	96
DIM	134	86
END	128	80
EXP	189	BD
FN	165	A5
FOR	129	81
FRE	184	B8
GET	161	A1
GO	203	CB
GOSUB	141	8D
GOTO	137	89
IF	139	8B
INPUT	133	85
INPUT#	132	84
INT	181	B5
LEFT$	200	C8
LEN	195	C3
LET	136	88
LIST	155	9B
LOAD	147	93

BASICBEFEHL	DEZ.-CODE	HEXCODE
LOG	188	BC
MID$	202	CA
NEW	162	A2
NEXT	130	82
NOT	168	A8
ON	145	91
OPEN	159	9F
OR	176	B0
PEEK	194	C2
POKE	151	97
POS	185	B9
PRINT	153	99
PRINT#	152	98
READ	135	87
REM	143	8F
RESTORE	140	8C
RETURN	142	8E
RIGHT$	201	C9
RND	187	BB
RUN	138	8A
SAVE	148	94
SGN	180	B4
SIN	191	BF
SPC(	166	A6
SQR	186	BA
STEP	169	A9
STOP	144	90
STR$	196	C4
SYS	158	9E
TAB	163	A3
TAN	192	C0
THEN	167	A7
TO	164	A4
USR	183	B7
VAL	197	C5
VERIFY	149	95
WAIT	146	92
↑	174	AE

Mit diesen Informationen, sowie den Codes für die CBM-ASCII-Darstellung sollte es Ihnen ohne weiteres möglich sein, Programmen etwas auf den Zahn zu fühlen und zu experimentieren, denn dadurch lernt man am meisten.

Uebersicht ueber die Speicherbelegung im C 64 mit Labels

HEX	Dezimal	FunKtion	Label
0000	0	Prozessorport-Datenrichtungsregister	D6510
0001	1	Input/Output-Register	R6510
0002	2	unbenuetzt	
0003-0004	3-4	Jump-Vector:Umwandlung FliessKomma->Fest	ADRAY1
0005-0006	5-6	Jump-Vector:Umwandlung Fest->FliessKomma	ADRAY2
0007	7	Such-CharaKter	CHARAC
0008	8	Flag:HochKomma-Stringende	ENDCHR
0009	9	Speicher:Spalte beim letzten Tab-Befehl	TRMPOS
000a	10	Flag: fuer Load (=0)/Verify (=1)	VERCK
000b	11	Zeiger: Basicworte (ToKenwandlung)	COUNT
000c	12	Flag: DIM	DIMFLG
000d	13	Datentyp: $ff=String/$00=numerisch	VALTYP
000e	14	Datentyp: $80=integer/$00=real	INTFLG
000f	15	Flag:Data-scan/List quote/Garbage coll	GARBFL
0010	16	Flag:User-call/Subscript-ref	SUBFLG
0011	17	Flag:Input (=$00)/Get (=$40)/Read (=$98)	INPFLG
0012	18	FLag:TAN-sign/Vergleichsergebnis	TANSGN
0013	19	Flag:Input-Prompt	?
0014-0015	20-21	Integer-Wert (z.B.:Zeilennummer)	LINNUM
0016	22	Pointer auf StringstacK	TEMPPT
0017-0018	23-24	Zeiger auf zuletzt benutzten String	LASTPT
0019-0021	25-33	StacK fuer Strings	TEMPST
0022-0025	34-37	Utility-Pointer-Bereich	INDEX
0026-002a	38-42	Speicher fuer Funktionsauswertung	RESHO
002b-002c	43-44	Zeiger: Start Basictext	TXTTAB
002d-002e	45-46	Zeiger: Start Basicvariable	VARTAB
002f-0030	47-48	Zeiger: Start Arrays	ARYTAB
0031-0032	49-50	Zeiger: Ende Arrays (+1)	STREND
0033-0034	51-52	Zeiger: Beginn Strings	FRETOP
0035-0036	53-54	Hilfszeiger fuer Strings	FRESPC
0037-0038	55-56	Zeiger: hoechste RAM-Adresse fuer Basic	MEMSIZ
0039-003a	57-58	laufende Basiczeilennummer	CURLIN
003b-003c	59-60	letzte Basiczeilen-Nummer	OLDLIN
003d-003e	61-62	Zeiger: naechstes Statement fuer cont	OLDTXT
003f-0040	63-64	laufende Data-Zeilen-Nummer	DATLIN
0041-0042	65-66	Zeiger: naechstes Data-Element	DATPTR
0043-0044	67-68	VeKtor: Input-Routine	INPPTR
0045-0046	69-70	Name der laufenden Basic-Variablen	VARNAM
0047-0048	71-72	Zeiger: Adresse der 1fd. Variablen	VARPNT
0049-004a	73-74	Zeiger: Indexvariable fuer FOR/NEXT	FORPNT
004b-004c	75-76	Zwischenspeicher fuer Programmzeiger	?
004d	77	MaSKe fuer Vergleichsoperationen	?
004e-004f	78-79	Zeiger fuer FN	?
0050-0053	80-83	Stringdescriptor	?
0054	84	Konstante: $4c (= jump) fuer Funktionen	?
0055-0056	85-86	Sprung-VeKtor fuer Funktionen	?
0057-005b	87-91	Register fuer ArithmetiK, AKKu 3	?
005c-0060	92-96	Register fuer ArithmetiK, AKKu 4	?
0061	97	FliessKomma-AKKu 1: Exponent	FACEXP
0062-0065	98-101	FliessKomma-AKKu 1: Mantisse	FACHO
0066	102	FliessKomma-AKKu 1: Vorzeichen	FACSGN
0067	103	Zeiger: Series Evalution Constant	SGNFLG
0068	104	FliessKomma-AKKu 1: Ueberlauf-Digit	BITS

0069	105	FliessKomma-AKKu 2: Exponent	ARGEXP
006a-006d	106-109	FliessKomma-AKKu 2: Mantisse	ARGHO
006e	110	FliessKomma-AKKu 2: Vorzeichen	ARGSGN
006f	111	Vorz.-Vergleichsergebnis AKKu 1 vs 2	ARISGN
0070	112	FliessKomma-AKKu 1: Rundungsbyte	FACOV
0071-0072	113-114	Zeiger: Polynomauswertung/Kassettenpuffer?	FBUFPT
0073-008a	115-138	Routine: holt naechst. Basictextbyte	CHRGET
0079	121	Einsprung: Zeichen nochmals holen	CHRGOT
007a-007b	122-123	Zeiger: laufendes Byte Basictext	TXTPTR
008b-008f	139-143	FliessKomma-Random-Wert	RNDX
0090	144	Kernal I/O-Status: Wort ST	STATUS
0091	145	Flag: Stop-Key/RVS-Key	STKEY
0092	146	ZeitKonstante fuer Kassette	SVXT
0093	147	Flag: Load (=$00)/Verify (=$01)	VERCK
0094	148	Flag: Serieller Bus Output (IEC-Output)	C3PO
0095	149	Ausgabepuffer Serieller Bus	BSOUR
0096	150	Kassetten-Sync-No.	SYNO
0097	151	Zwischenspeicher Daten	?
0098	152	Anzahl offene Files/Index fuer Filetabelle	LDTND
0099	153	aKtives Eingabegeraet (Standard: 0)	DFLTN
009a	154	aKtives Ausgabegeraet (Standard: 3)	DFLTO
009b	155	Parity fuer Kassette	PRTY
009c	156	Flag: Kassette-Byte empfangen	DPSW
009d	157	Flag: Direktmodus ($80)/Programm ($00)	MSGFLG
009e	158	Kassette Pass 1 Error Log	PTR1
009f	159	Kassette Pass 2 Error Log	PTR2
00a0-00a2	160-162	Real-Time-Jiffy ClocK ca. 1/60 sec	TIME
00a3	163	Bitzaehler serielle Ausgabe	?
00a4	164	Zaehler fuer Kassette	?
00a5	165	Kassetten-Sync-Zaehler	CNTDN
00a6	166	Zeiger: Tape I/O-Buffer	BUFPNT
00a7	167	RS 232 Input Bits/Kassette	INBIT
00a8	168	RS 232 Input Bit-Zaehler/Kassette	BITCI
00a9	169	Flag: RS 232 ChecK Startbit	RINONE
00aa	170	RS 232 Input-Byte-Puffer	RIDATA
00ab	171	RS 232 Input-Parity/Kassetten Short-Zaehler	RIPRTY
00ac-00ad	172-173	Zeiger: Bandpuffer und Scrolling	SAL
00ae-00af	174-175	Zeiger: Programmende (Load/Save-Kassette)	EAL
00b0-00b1	176-177	ZeitKonstanten fuer Kassettenbetrieb	CMP0
00b2-00b3	178-179	Zeiger: Start Tape-Buffer	TAPE1
00b4	180	RS 232 Out-Bit-Count/Kassette	BITTS
00b5	181	RS 232 Naechstes Ausgabebit/EOT-Flag	NXTBIT
00b6	182	RS 232 Ausgabe-Byte-Puffer	RODATA
00b7	183	Laenge des 1fd. Filenamens	FNLEN
00b8	184	Laufende logische File-Nummer	LA
00b9	185	Laufende SeKundaeradresse	SA
00ba	186	Laufende Geraete-Nummer	FA
00bb-00bc	187-188	Zeiger auf laufenden File-Namen	FNADR
00bd	189	RS 232 Ausgabe-Parity/Kassette	ROPRTY
00be	190	BlocKzaehler fuer Band	FSBLK
00bf	191	Serieller Word-Puffer	MYCH
00c0	192	Flag fuer Kassettenmotor	CAS1
00c1-00c2	193-194	I/O-Start-Adresse	STAL
00c3-00c4	195-196	Endadresse fuer Ein-/Ausgabe	MEMUSS
00c5	197	gedrueckte Taste (0=Keine Taste gedrueckt)	LSTX
00c6	198	Anzahl gedrueckter Tasten	NDX
00c7	199	Flag: RVS	RVS

00c8	200	Zeiger: Zeilenende fuer Eingabe	INDX
00c9-00c9	201-202	Cursor x-y-Position fuer Eingabe	LXSP
00cb	203	Flag: Ausgabe geshiftetes Zeichen	?
00cc	204	Flag: Cursor ein (0) / aus (1)	?
00cd	205	BlinKzaehler fuer Cursor	?
00ce	206	Zeichen unter em Cursor	?
00cf	207	Flag: Cursor-Phase (1=ein / 0=aus)	?
00d0	208	Flag: Eingabe Tastatur oder Bildschirm	?
00d1-00d2	201-210	Zeiger: aKtuelle Bildschirmzeile (Start)	?
00d3	211	Cursorspalte	?
00d4	212	Flag: HochKomma-Modus	?
00d5	213	Laenge der Bildschirmzeile	?
00d6	214	Cursorzeile	?
00d7	215	diverse ZwecKe	?
00d8	216	Anzahl der Inserts	?
00d9-00f2	217-242	MSB der Bildschirmanfaenge	?
00f3-00f4	243-244	Zeiger: in Farb-RAM	?
00f5-00f6	245-246	Zeiger: TastaturdeKodiertabelle	?
00f7-00f8	247-248	Zeiger: RS232-Eingabepuffer	?
00f9-00fa	249-250	Zeiger: RS232-Ausgabepuffer	?
00fb	251	unbenutzt	
00fc	252	unbenutzt	
00fd	253	unbenutzt	
00fe	254	unbenutzt	
00ff-010f	255-271	Puffer: Umwandlung FliessKomma -> ASCII	?
0100-017b	256-379	Speicher: BandKorreKtur (Adr. fehlerh. Bytes)	
0100-01ff	256-511	Prozessor-StacK	
01fc-01ff	508-511	Puffer: bei Einbau neuer Basic-Zeilen	
0200-0258	512-600	Basiceingabepuffer	BUF
0259-0262	601-610	Tabelle: logische File-Nummern	LAT
0263-026c	611-620	Tabelle: Geraete-Nummern	FAT
026d-0276	621-630	Tabelle: SeKundaeradressen	SAT
0277-0280	631-640	Tastaturpuffer (FIFO)	KEYD
0281-0282	641-642	Zeiger: Start Basic-RAM	MEMSTR
0283-0284	643-644	Zeiger: Ende Basic-RAM	MEMSIZ
0285	645	Flag: Time-out serieller Bus	TIMOUT
0286	646	aKtuelle Zeichen-Farbe	COLOR
0287	647	Hintergrundfarbe unter dem Cursor	GDCOL
0288	648	High-Byte-Video-RAM (=Page)	HIBASE
0289	649	Laenge des Tastaturpuffers	XMAX
028a	650	Flag: RepeatfunKtion	RPTFLG
028b	651	Zaehler: Repeat-GeschwindigKeit	KOUNT
028c	652	Zaehler: Repeat-Verzoegerung	DELAY
028d	653	Flag: Shift/Commodore/Control (Bit 0,1,2)	SHFLAG
028e	654	Flag: Shift letztes Muster	LSTSHF
028f-0290	655-656	VeKtor: Tastatur-DeKodierung	KEYLOG
0291	657	Flag: Shift/Commodore (0=gesperrt,$80 frei)	MODE
0292	658	Flag: Auto-Scroll down (0=ein)	AUTODN
0293	659	RS 232: 6551 Kontrol-Register	M51CTR
0294	660	RS 232: 6551 Kommando-Register	M51CDR
0295-0296	661-662	RS 232: Wert fuer Baudrate aus Tabelle	M51AJB
0297	663	RS 232: 6551 Status-Byte	RSSTAT
0298	664	RS 232: Anzahl der zu sendenden Bits	BITNUM
0299-029a	665-666	RS 232: Timer beim Senden	BAUDOF
029b	667	RS 232: Index auf Ende Empfangspuffer	RIDBE
029c	668	RS 232: Beginn Input-Puffer (Page)	RIDBS
029d	669	RS 232: Beginn Output-Puffer (Page)	RODBS

029e	670	RS 232: Index auf Ende Output-Puffer	RODBE
029f-02a0	671-672	Speicher fuer IRQ waehrend Kassetten-I/O	IRQTMP
02a1	673	RS 232: CIA 2 NMI-Flag (RS 232 Enables)	ENABL
02a2	674	CIA 1 Timer A/ TOD-Sense during Tape-I/O	?
02a3	675	CIA 1 Interuptflag	?
02a4	676	CIA 1 Flag fuer Timer A	?
02a5	677	Speicher fuer Zeilenindex	?
02a6	678	Flag: Quarzfrequenzen (Pal =1, NTSC =0)	
02a7-02bf	679-703	unbenutzt ?	?
02c0-02fe	704-766	Sprite 11	?
02ff	767	unbenutzt ?	?
0300-0301	768-769	VeKtor: Basic-Fehler-Meldung oder Fehlermeldung in AKKu, dann Warmstart	IERROR
0302-0303	770-771	VeKtor: Eingabe einer Basiczeile	IMAIN
0304-0305	772-773	VeKtor: Basic-ToKen-Umwandlung	ICRNCH
0306-0307	774-775	VeKtor: Umwandlung in Klartext (List)	IQPLOP
0308-0309	776-777	VeKtor: Basic-Befehlsadresse	IGONE
030a-030b	778-779	VeKtor: ToKen Umwandlung	IEVAL
030c	780	Speicher: AKKu	SAREG
030d	781	Speicher: X-Register	SXREG
030e	782	Speicher: Y-Register	SYREG
030f	783	Speicher: StacKpointer (Stapelzeiger)	SPREG
0310	784	jmp-Befehl fuer USR-Funktion ($4c)	USRPOK
0311-0312	785-786	USR-Adresse (LB/HB)	USRADD
0313	787	unbenutzt	?
0314-0315	788-789	VeKtor: Hardware-IRQ	CINV
0316-0317	790-791	VeKtor: BRK	CBINV
0318-0319	792-793	VeKtor: NMI	NMINV
031a-031b	794-795	VeKtor: Open-Routine	IOPEN
031c-031d	796-797	VeKtor: Close-Routine	ICLOSE
031e-031f	798-799	VeKtor: CHKIN (Eingabegeraet setzen)	ICHKIN
0320-0321	800-801	VeKtor: CKOUT (Ausgabegeraet setzen)	ICKOUT
0322-0323	802-803	VeKtor: CLRCH (aKtiven I/O-Kanal schliessen)	ICLRCH
0324-0325	804-805	VeKtor: Input (BASIN = Zeicheneingabe)	IBASIN
0326-0327	806-807	VeKtor: Output (BSOUT =Zeichenausgabe)	IBSOUT
0328-0329	808-809	VeKtor: Stop (Stoptaste abfragen)	ISTOP
032a-032b	810-811	VeKtor: Getin ($f13e)	IGETIN
032c-032d	812-813	VeKtor: CLALL (alle I/O-Kanaele schliessen)	ICLALL
032e-032f	814-815	VeKtor: Warmstart bzw. User-defined	USRCMD
0330-0331	816-817	VeKtor: Load	ILOAD
0332-0333	818-819	VeKtor: Save	ISAVE
0334-033b	820-827	unbenutzt ??	?
033c-03fb	828-1019	Kassettenpuffer	TBUFFR
0340-037e	832-894	Sprite 13	
0380-03be	896-958	Sprite 14	
03c0-03fe	960-1022	Sprite 15	
03fc-03ff	1020-1023	unbenutzt	?
0400-07e7	1024-2023	Video-Matrix: 25 Zeilen je 40 Zeichen	VICSCN
07f8-07ff	2040-2047	Sprite-Daten-Zeiger	
0800-9fff	2048-40959	Basic-Bereich normal	
8000-9fff	32768-40959	EinstecK-Module (8192 Bytes)	
a000-bfff	40960-49151	Basic-ROM (8192 Bytes)	
c000-cfff	49152-53247	freies RAM (4096 Bytes)	
d000-dfff	53248-57343	I/O-Devices und Color-RAM oder CharaKter-Generator-ROM oder RAM (4096 Bytes)	
e000-ffff	57344-65535	Kernal-ROM oder RAM (8192 Bytes)	

Verschiedenes

Verschiedene Peeks, Pokes und Tricks

Autostart von Diskettenprogrammen:
Eingabe:
Load"name",8:(Shift+Run/Stop-Taste)

Lese-/Schreib-Töne des Kassettenrecorders kann man durch Poke54296,15 hörbar machen.

Sys64738 (=$fce2) bewirkt Systemreset.

Sys65499 (=$ffdb) setzt TI$ auf "000000"

Poke199,1 bewirkt, daß der nachfolgend auszugebende Text Revers erscheint, also poke199,1:print"ComputerSchau".

SYS65126 ($fe66) bewirkt, daß die Standardvektoren für I/O und für die Interrupts wieder gestellt werden. Danach wird der Bildschirm gelöscht und zum Basic-Warmstart gesprungen.

Sys44808 ($af08) gibt ?syntax error aus.

---

sys58260 ($e394) bewirkt Basic-Kaltstart.

---

poke780,158:poke782,160:sys43806 gibt die Basicbefehlsworte aus.
Wie Sie leicht sehen können, ist bei den letzten Buchstaben immer das höchste Bit gesetzt. Am besten schalten Sie in den Kleinschriftmodus um.
Oder
poke780,116:poke782,228:sys43806 gibt den String „★★★★ commodore 64 basic v2 ★★★★", danach zweimal Cursor down und dann den String „64K ram system" aus. Falls Sie ein dokumentiertes ROM-Listing des C 64 besitzen, werden Sie unschwer aus den beiden Beispielen erkennen können, daß in 780 das „lower Byte" und in 782 das Hi-Byte der Adresse, in welcher der String zu finden ist, geschrieben werden muß. Das Ende der Ausgabe erfolgt durch ein „00"-Byte.

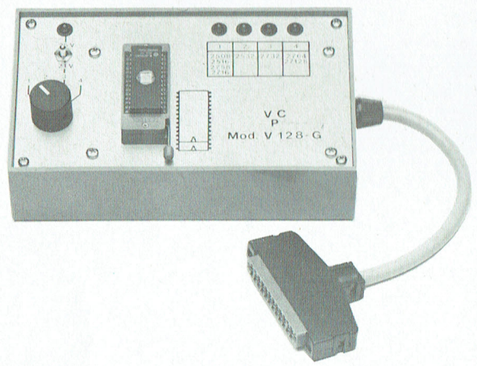
Durch Poke 650,255 wird eine Repeatfunktion für alle Tasten eingeschaltet. Die Rückstellung erfolgt durch poke650,0.

Das Abschalten der Dauerfunktion aller Tasten kann durch Poke650,127 erfolgen.

---

poke657,128 stellt Umschaltung Groß-/Kleinschrift ab.
poke788,49 unterbindet Funktion der Stoptaste.
poke649,0 keine Eingaben über Tastatur mehr möglich, da Tastaturpuffer auf 0 gesetzt wurde.
printpeek(60)★256+peek(59) gibt die letzte Zeilennummer aus, die bearbeitet wurde.
Sys65511 schließt alle offenen Files.

V C P | Völzke Computer Peripherie | V C P



Eprom-Programmer V128 für VC-20, C-64 u. SX-64; für Eproms 2508/16/32 und 2758/16/32/64/128. Professionelle Ausführung mit erweiterter Treiber-Software auf Kassette: **DM 249,-**

Eprom-Programmer V128-G
im Pult-Gehäuse, siehe Abbildung oben: **DM 349,-**

UNIMENT, C-64-Befehlserweiterung mit über 50 zus. Befehlen u. Funktionen für Assembler, Centronics-Druckeranschluß, Graphik-, Sprite-, Sound- und Disketten-Anwendungen; mit Beispielprogrammen und ausführlicher Bedienungsanleitung: **DM 99,-**

UNIMENT-Autostart-Steckmodul: **DM 199,-**

BASIC-Burner erlaubt das Schießen von Basic-Programmen als Autostart-Moduln in Eproms. Diese Software wurde speziell für unsere Eprom-Programmer entwickelt. Kassette: **DM 99,-**

Außerdem liefern wir Eprom-Karten u. -Löschgeräte, 80-Zeichenkarten u.v.a.m. Fordern Sie kostenloses Info gegen Rückporto an.

Wir liefern ab Lager Pullach. Alle Preise sind inkl. 14% MWSt.

Hagen Völzke, Ahornallee 4, 8023 Pullach
Versandhandel Tel.: 089/793 4534

So beherrschen Sie den Commodore 64!

nur 29,80

Die vielfältigen Möglichkeiten des Commodore 64.

Mit SIMON'S BASIC.

Eine Einführung für den Einsteiger.

Anregungen und Tips für den Anwender.

Nach kurzer Zeit bereits in 2. Auflage!

Kaum auf dem Markt, eroberte sich der Mikrocomputer „Commodore 64" im Sturm die Anwendungsbereiche Arbeitsplatz, Unterricht und Freizeit.

Mit dem Commodore 64 läßt sich eine Menge machen. Doch welche Möglichkeiten Ihnen als Anwender insgesamt zur Verfügung stehen, erfahren Sie erst, wenn Sie das neue Handbuch gelesen haben: „Commodore 64, Daten, Text, Grafik, Musik."
Anerkannte Fachleute schreiben hier über alles, was Sie über den Commodore 64 wissen müssen.

Auch wer erst jetzt einsteigen will und vor dem Kauf eines Mikrocomputers steht, sollte dieses Buch lesen.

Von Diplom-Physiker Dr. Hans Riedl und Diplom-Kaufmann Franz Quinke
2. Auflage.
1984. 164 Seiten.
DM 29,80
ISBN 3 470 **80422** 2

Kiehl Verlag
6700 Ludwigshafen

Endlich wird es ganz einfach, mit Computern und Programmen umzugehen:

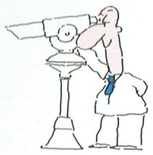
Denn die „ComputerSchau" zeigt, was Sie mit den auf dem Markt befindlichen Geräten alles machen können.

Die „ComputerSchau" zeigt, wo man als Anwender oder zukünftiger Anwender alles findet, was man braucht.

„ComputerSchau" ist im Handel und kostet DM 6,–.
Sie werden feststellen, daß sie diesen Preis wert ist – schon nach der Lektüre des ersten Artikels.

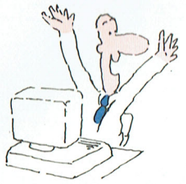
Die „ComputerSchau" zeigt, nach welchen Kriterien Sie Ihren Computer aussuchen sollten.

Die „ComputerSchau" zeigt, wie Sie Ihren Computer und das Drumherum optimal nutzen.
Die „ComputerSchau" zeigt, wie Sie aus preiswerten Standardprogrammen wertvolle Individualprogramme machen.

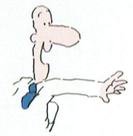


Die „ComputerSchau" zeigt, was andere mit ihrem Computer alles anstellen.

Die „ComputerSchau" zeigt, mit welchen Kniffen Sie verzwickte Probleme lösen.



Die „ComputerSchau" zeigt, wie Sie noch mehr Spaß beim Spiel mit Ihrem Computer bekommen.



Computer Schau
zeigt, wie man's macht.

SIE können von unserem Kennenlern-Angebot Gebrauch machen.
Die Karte dafür finden Sie nebenstehend.